Randomized Search of Graphs in Log Space and Probabilistic Computation

Wen-Ju Cheng
Graduate Center, City University of New York

10-1-2014

Recommended Citation
http://academicworks.cuny.edu/gc_etds/411

How does access to this work benefit you? Let us know!
Follow this and additional works at: http://academicworks.cuny.edu/gc_etds
Part of the Computer Sciences Commons

This Dissertation is brought to you by CUNY Academic Works. It has been accepted for inclusion in All Graduate Works by Year: Dissertations, Theses, and Capstone Projects by an authorized administrator of CUNY Academic Works. For more information, please contact deposit@gc.cuny.edu.
RANDOMIZED SEARCH OF GRAPHS IN LOG SPACE
AND PROBABILISTIC COMPUTATION

by

WEN-JU CHENG

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2014
This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

James L. Cox

_________________________________________  Chair of Examining Committee

Robert Haralick

_________________________________________  Executive Officer

Stathis Zachos

Nelly Fazio

Noson S. Yanofsky  Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK
Abstract

RANDOMIZED SEARCH OF GRAPHS IN LOG SPACE
AND PROBABILISTIC COMPUTATION

by

Wen-Ju Cheng

Advisor: Professor James Cox

Reingold has shown that L = SL, that s-t connectivity in a poly-mixing digraph is complete for promise-RL, and that s-t connectivity for a poly-mixing out-regular digraph with known stationary distribution is in L. Several properties that bound the mixing times of random walks on digraphs have been identified, including the digraph conductance and the digraph spectral expansion. However, rapidly mixing digraphs can still have exponential cover time, thus it is important to specifically identify structural properties of digraphs that effect cover times. We examine the complexity of random walks on a basic parameterized family of unbalanced digraphs called Strong Chains (which model weakly symmetric logspace computations), and a special family of Strong Chains called Harps. We show that the worst case hitting times of Strong Chain families vary smoothly with the number of asymmetric vertices and identify the necessary condition for non-polynomial cover time. This analysis also yields bounds on the cover times of general digraphs.

Next we relate random walks on graphs to the random walks that arise in Monte Carlo methods applied to optimization problems. We introduce the notion of the asymmetric states of Markov chains and use this definition to obtain some results about Markov chains. We also obtain some results on the mixing times for Markov Chain Monte Carlo Methods.
Finally, we consider the question of whether a single long random walk or many short walks is a better strategy for exploration. These are walks which reset to the start after a fixed number of steps. We exhibit digraph families for which a few short walks are far superior to a single long walk. We introduce an iterative deepening random search. We use this strategy estimate the cover time for poly-mixing subgraphs. Finally we discuss complexity theoretic implications and future work.
Acknowledgements

I would like to thank my committee members, colleagues, and family for their help, support, and guidance in this work. First of all I want to express my special appreciation and thanks to my advisor Professor Cox for the continuous support of my study and for his patience, motivation, enthusiasm, and immense knowledge, and to Professor Zachos for his guidance and encouragement. I acknowledge and thank the other members of my Thesis Advisory and Defense Committee: Professor Nelly Fazio and Professor Noson Yanofsky for their support and encouragement during the course of my work.

I want to thank all of my family. I could not have completed this work without their constant support and encouragement. Finally, I give special thanks and love to my daughter Cindy.
## Contents

1. **Introduction** 01

2. **Preliminaries** 07

3. **Relevant Prior Work** 13
   - 3.1 Electric Networks and Cover Time 13
   - 3.2 Circulation and directed Cheeger Constants 16
   - 3.3 $\text{SL} = L$ and Promise-RL 19
   - 3.4 $\text{ReachFewL} = \text{ReachUL}$ 24

4. **Weakly Symmetric Classes** 28
   - 4.1 Strong Chains 29
   - 4.2 Strong Chain Symmetry 31
   - 4.3 Symmetric and Asymmetric Graphs 32
   - 4.4 Categorizing Strong Chains 47
   - 4.5 Parameterized Complexity of Strong Chains 49

5. **Markov Chain Monte Carlo (MCMC) methods** 53
   - 5.1 Degree of Balance 57
   - 5.2 Markov Chain Monte Carlo (MCMC) Methods 59
5.2.1 The \( n \)-vertex digraph \( D \) of a Probabilistic Turing Machine 60

5.2.2 Transforming \( D \) to a \( 4^t \)-1-vertex complete 4-tree \( D' \) 61

5.2.3 \( \kappa \)-vertex Sample Set \( S \) 62

6 Many Short Walks Can Be Better Than One Long Walk 66

6.1 Layer Digraphs 67

6.2 Reachability 70

6.2.1 \( s-k \)-reachable set \( (R) \) 70

6.2.2 Poly-Reachability 72

6.3 Directed Cheeger Constants 74

6.4 Bagging Vertices 75

7 Conclusion and Future Work 77

Bibliography 81
List of Figures

Figure 2.1 Relations of Complexity Classes 09
Figure 3.1 Graph as an electrical network 14
Figure 3.2 An example of digraph has exponential cover time 17
Figure 3.3 Replacement by cycle 21
Figure 3.4 An example of zig-zag product 22
Figure 3.5 (a) Graph G (b) The corresponding 3-page graph H. The dashed edges are on the page 3. 26
Figure 4.1 An example of Strong Chain 30
Figure 4.2 Harp Strings Digraph 32
Figure 4.3 Line Digraph 33
Figure 4.4 Eulerian Path Digraph 34
Figure 4.5 Eulerian Cycle Digraph 35
Figure 4.6 2-regular digraph 35
Figure 4.7 Line digraph modification $L_1$ 36
Figure 4.8 Line digraph modification $L_2$ 41
Figure 4.9 Mirror Frame digraph 43
Figure 4.10 Mirror Frame digraph 45
Figure 4.11 An example of a simple Harp 48
Figure 6.1 An example of a 4-layer digraph 67
Figure 6.2 Graph Family 1, $F_1$ 68
Figure 6.3: Graph Family 2, $F_2$ 69
List of Algorithms

Algorithm 6.1 Random Walk($G, s, t, \ell$) 70
Algorithm 6.2 Inductive Search($G, s, t$) 71
Chapter 1

Introduction

Recall that L (respectively NL) is the class of languages accepted by deterministic (respectively nondeterministic) log space bounded Turing machines (space bounded by $O(\log n)$). Recall also that RL is the class of languages accepted by probabilistic log space bounded Turing machines with one-sided error. It is well known that STCON, the problem of determining if there exists a path between two specified vertices, s and t, in a given directed graph is complete for NL. Symmetric Logspace (SL) was first introduced by Lewis and Papadimitriou in 1982 [LP]. They defined SL (symmetric log space) as the class of all languages accepted by a log space bounded symmetric Turing machine. USTCON is the restriction of STCON to undirected graphs. They showed that USTCON is complete for SL, and thus proved that $L \subseteq SL \subseteq NL$, since it was not at the time how to search an undirected graph in deterministic log space. Aleliunas et. al. ([AKLLR], 1979) introduced a randomized log-space algorithm for USTCON. They demonstrated that a random walk on an undirected graph will reach the target with high probability in polynomial time. This shows that $USTCON \in RL$. Savitch’s theorem ([Savitch], 1970) can be used to show that $NL \subseteq L^2$ (deterministic $O(\log^2 n)$ space) by demonstrating a simulation of a non-deterministic space $S$ machine by a deterministic space $S^2$ machine. Nisan, Szemeredi, and Wigderson ([NSW], 1992) showed that USTCON is in $L^{3/2}$, thus showing $SL \subseteq L^{3/2}$. Saks and Zhou ([SZ], 1995) showed the stronger result that any randomized space $S$ machine can be simulated by a deterministic space $S^{3/2}$ machine, implying that $RL \subseteq L^{3/2}$. In
1997, Armoni et. al. improved this deterministic simulation to give a $O(\log^{4/3} n)$-space algorithm for USTCON. The relation between SL and L had remained open until 2005, when Reingold showed the surprising result that any undirected graph can be searched deterministically in log space. The completeness of USTCON for SL and the fact that it is now known to be in L thereby establishes that $SL = L$. Summarizing the above facts we now know that:

$$L = SL \subseteq RL \subseteq NL = Co-NL \subseteq L^2$$

and

$$RL \subseteq L^{3/2}$$

Reingold showed how to transform any graph (using log space) into an expander with constant degree. Since the transformed graph has constant degree, we can perform the search by systematically enumerating all paths of length $O(\log n)$, as edge sequences, using $O(\log n)$ space. Rozenman and Vadhan ([RV], 2005) showed how to accomplish this transformation without increasing the size of the graph significantly (by much less than the blowup in Reingold’s technique) by the use of pseudorandomized graph squaring, an operation that increases the expansion of the graph, yet only increases the degree by a constant. This method also employs a sequence of expander graphs, essentially relying on the statistical properties of random walks on expanders (which converge quite rapidly to a nearly uniform distribution on the vertices) that have proven quite useful in pseudorandomness.

Research in pseudorandomness aims to construct “random like” objects using little or no randomness. Fooling space bounded computations has established some relations between probabilistic and deterministic space bounded classes. In some early work pseudo-generators were constructed using families of universal and perfect hash functions, such as those defined by Carter and Wegman ([CW], 1979). Recently many results use the fact that $2\log n$ random bits
can be generated by a truly random seed of length $\log n + \log k$ on a $k$-expander graph. One selects a random vertex $v$ with $\log n$ bits then selects a neighbor of $v$ with $\log k$ bits. This process provides statistically good coverage of the combinatorial rectangle defined by all pairs of vertices. This can be generalized recursively to generate $O(\text{poly}(n))$ pseudorandom bits using $O(\log^2 n)$ random bits through a modification of Nisan’s ([Nisan], 1992) space fooling generator, that replaces hash functions by expanders ([INW], 1994). The close relationship among expanders, hash functions, and extractors has now been well established (see for example [Vadhan], 2012). Motivated by recent work of Reingold’s and others ([CRV], [BRZ], [De], [GMRZ], [LZ10], and [LZ11]), much research on pseudorandomness (and the related topic of traversal sequences) now employs expander graphs.

The pseudo-generators of Nisan [Nissan] and Impagliazzo, Nisan, and Wigderson [INW] require $O(\log^2 n)$ random bits to fool a log space bounded machine. In addition to complexity theoretic implications, the improvement of the INW generator would have many practical applications. However, the random bits required by the above have not improved much in almost two decades. The best known result is the $O(\log m + \log n + \log^2(1/\varepsilon))$ pseudorandom generator for combinatorial shapes [GMRZ]. A random seed of length $O(\log n (\log \log n) + \log(1/\varepsilon))$ has been used by a number of researchers (see, for example, [De]) to fool constant width regular branching programs. Pseudo-generators with seed length $O(\log n)$ for group products were introduced by [LRTV] and [MZ], but they only work for a restricted class of digraphs. Reingold, et. al ([LRTV], 2009) use a sequence of expander graphs (Cayley graphs) to construct a pseudorandom generator that fools modular sums.

Research on extending Reingold’s original algorithm to the directed case has been undertaken by Chung, Reingold and Vadhan ([CRV], 2011), in an attempt to prove $\text{RL} = \text{L}$.
Their best result shows that STCON on a poly-mixing digraph with a known stationary distribution is in L. They show that STCON on a poly-mixing digraph is complete for Promise-RL. The complexity of searching such a digraph, without knowing the stationary distribution, remains open. This, and similar recent work, has primarily focused on analyzing the search problem for either general digraphs or digraphs with a restricted structure so that they “look like” undirected graphs.

Several known graph properties have been extended to digraphs: the spectral expansion (or alternatively the spectral gap) and the digraph conductance. The former property is a linear algebraic property of a suitably defined digraph Laplacian, while the latter property is a combinatorial or structural property of the digraph. These properties provide bounds on the rate of convergence of random walks on the digraphs to the stationary distribution, if it exists. However, as we observe below, these properties give only very weak lower bounds on the cover time (the expected time to visit all vertices) and cannot distinguish between polynomial mixing times (fast convergence) and exponential cover times (slow exploration).

In a slightly different direction, we are interested in examining complexity of the NL class. If NL is not equal to L, and if RL = L, as is generally believed, then there is a large gap between RL and NL. We aim to categorize the problem of STCON on families of strongly connected digraphs that do not have polynomial cover times, which will correspond to different (potential) sub-NL complexity classes. Our goal is to produce a hierarchy of computation graphs that potentially lie between RL and NL.

We define a parameterized family of digraphs, called Strong Chains, which will represent some simple nondeterministic computations. We investigate random walks on these classes. We analyze the structure of the family of digraphs by introducing several notions of degree of
symmetry. Related work by Li and Zhang ([LZ11], 2011) has applied a notion of asymmetry to digraphs, by relating it to the probability assigned to the edges of an essentially regular digraph. In contrast, our definitions are related to the actual digraph structure. We then define another parameterized family of special Strong Chains called Harps, which are in a sense “complete” for strong chains. For given parameters, the hitting time of a random walk on the family of Harps will dominate the corresponding family of Strong Chains. Since the Harps have a very simple structure that embodies the asymmetry of families of Strong Chains, this enabled us to identify a structural property of digraphs that leads to search/random walk complexity. The property is the number of asymmetric vertices. The property enables us to obtain bounds on the cover times of Strong Chains and extend these bounds to general digraphs. The number of asymmetric vertices provides better bounds than either the digraph conductance or the digraph spectral expansion. These latter two measures provide bounds on the rate of convergence of a random walk to the stationary distribution, and hence the mixing time. Nevertheless, a rapidly mixing digraph may still have exponential cover time, and these measures fail to identify this fact. By contrast the number of asymmetric vertices is a parameter that enables us to identify a necessary condition for the digraph to have superpolynomial cover time. We use construct a stratification of Strong Chains using this parameter, and identify a class of digraphs that we conjecture have intermediate complexity between RL and NL.

Randomized graph search has a deep relation to probabilistic complexity classes, as we have noted. We can thus apply both previously known results and our new work to the analysis of probabilistic computation. We relate random walks on graphs to the random walks that arise in Monte Carle methods applied to optimization problems. We introduce the notion of the asymmetric states of a Monte Carlo process and obtain some results for the corresponding
Markov chains. The results of the cover time for directed graphs can also be modified to obtain bounds on the mixing time for certain Markov Chain Monte Carlo (MCMC) methods applied to discrete problems domains.

Finally, we consider a more practical problem: exploring an unknown neighborhood while using limited space and time. We study the search of weakly connected digraphs by using many short random walks. These are walks which reset to the start after a fixed number of steps. We are interested in when a long random walk may be replaced by many short walks. We identify digraph families for which this strategy is more effective, with a comparable time bound, than a single random walk of long duration. We are expanding our study of Strong chains to determine the error bound as a function of the duration of the short walk. We are also interested in the situation in which the many short walks strategy can more effectively visit the subset of vertices which have both non-negligible probability and rapid convergence times. We estimate the cover time for these poly-mixing subgraphs, using a kind of iterative deepening random search.
Chapter 2

Preliminaries

In this dissertation we will mainly focus on all the logspace complexity classes. We define all the complexity classes we use below. In the following a logarithmic space means working memory bounded by $O(\log n)$, for input of size $n$ bits.

2.1 $L$ (Logarithmic Space): is the complexity class containing decision problems that can be solved by a deterministic Turing machine using logarithmic space.

2.2 $SL$ (Symmetric Logspace): is the complexity class containing decision problems that can be solved by a nondeterministic Turing machine in logarithmic space, such that

1. If the answer is 'yes,' one or more computation paths accept.
2. If the answer is 'no,' all paths reject.
3. If the machine can make a nondeterministic transition from configuration $A$ to configuration $B$, then it can also transition from $B$ to $A$. (This is what 'symmetric' means.)

2.3 $NL$: The complexity class containing decision problems that can be solved by a nondeterministic Turing machine using a logarithmic amount of memory space.

Reingold has shown that $L = SL$. The logspace bounded classes that contain the class $L$, are defined by either nondeterminism or randomization. The nondeterministic classes include the classes $SL$ and $NL$, and they also include certain “semantic” classes, which are defined by some bound on the number of computation paths, e.g. $ReachUL$. These classes are known to be
subsets of $NL$. Logspace classes are also defined by randomization. The randomized classes $BPL$ and $PL$ are not known to be subsets of $NL$. We continue with the definition of the randomized classes.

2.4 $RL$ (Randomized Logspace): The class of decision problems solvable by a non-deterministic Turing machine in logarithmic space, such that

1. If the answer is 'yes,' at least 1/2 of computation paths accept.
2. If the answer is 'no,' all computation paths reject.

2.5 $BPL$ (Bounded-Error Probabilistic Logspace): The class of decision problems solvable by a nondeterministic Turing machine in logarithmic space, such that

1. If the answer is 'yes,' at least 2/3 of computation paths accept.
2. If the answer is 'no,' at most 1/3 of computation paths accept.

2.6 $PL$ (Probabilistic Logspace): The class of decision problems solvable by a non-deterministic Turing machine, such that

1. If the answer is 'yes' then at least 1/2 of computation paths accept.
2. If the answer is 'no' then less than 1/2 of computation paths accept.

We now define the following semantic logspace classes.

2.7 $UL$ (Unambiguous Logspace): The class of decision problems solvable by a non-deterministic Turing machine in logarithmic space, such that

1. If the answer is 'yes,' exactly one computation path accepts.
2. If the answer is 'no,' all computation paths reject.

2.8 $FewL$: The class of decision problems solvable by a non-deterministic Turing machine in logarithmic space, such that
1. If the answer is 'yes,' there is at least one accepting path and at most a polynomial number of computation paths accept.

2. If the answer is 'no,' all computation paths reject.

A nondeterministic machine is said to be reach-unambiguous if for every pair of configurations that are reachable from the start configuration, there exists at most one computational path connecting these configurations.

2.9 ReachUL is the class of decision problems solvable by a reach-unambiguous non-deterministic Turing machine in logarithmic space with a unique accepting path.

2.10 ReachFewL was originally defined as the class of decision problems solvable by a reach-unambiguous non-deterministic Turing machine in logarithmic space with at most a polynomial number of computation paths. It was recently shown that ReachFewL = ReachUL.
In the following we will use $\text{poly}(n)$ to denote any arbitrary polynomial function of $n$ and $\text{exp}(n)$ will denote any exponential function of $n$.

A directed graph (or digraph) $G$ is a pair $(V, E)$, where $V$ is a finite set and $E$ is a binary relation on $V$. The set $V$ is called the vertex set of $G$, and its elements are called vertices. The set $E$ is called the edge set of $G$, and its elements are called edges. In an undirected graph $G = (V, E)$, the binary relation $E$ is symmetric so that the edge set $E$ consists of unordered pairs of vertices. We use the notation $(u, v)$ for an edge. If $(u, v)$ is an edge in a digraph $G$, we say that $(u, v)$ leaves vertex $u$ and is enters vertex $v$, and that $u$ and $v$ are adjacent.

The degree of a vertex in an undirected graph is the number of edges incident upon it. In a digraph, the out-degree of a vertex is the number of edges leaving it, and the in-degree of vertex is the number of edges entering it.

A path of length $k$ from a vertex $u$ to a vertex $w$ in a graph $G = (V, E)$ is a sequence $\{v_0, v_1, v_2, \ldots, v_k\}$ of vertices such that $u = v_0$, $w = v_k$, and $(v_{i-1}, v_i) \in E$ for $i = 1, 2, \ldots, k$. The length of the path is the number of edges in the path. If there is a path $p$ from $u$ to $w$, we say that $w$ is reachable from $u$ via $p$. A path is simple if all vertices in the path are distinct.

An undirected graph is connected if every pair of vertices is connected by a path. A directed graph is strongly connected if every two vertices are reachable from each other. In an undirected graph $G = (V, E)$, $u$ and $v$ are neighbors if they are adjacent.

Let $G = (V, E)$ be a graph with $n$ vertices and $m$ edges. A simple random walk on $G$ is a sequence of discrete steps starting at a vertex $v_0$, so that at each step $i$, a neighbor of the current vertex $v_i$ is chosen uniformly at random as the next vertex $v_{i+1}$. A $\frac{1}{2}$ lazy random walk is a walk that chooses stay at the current vertex, i.e. rejects the move, with probability $\frac{1}{2}$.
2.11 A Markov Chain $M$ is a discrete-time stochastic process defined over a set of states $S$ in terms of a matrix $P$ of transition probabilities. The Markov chain is in one state at any time, making state-transitions at discrete time-steps $t = 1, 2, \ldots$. The entry $P_{ij}$ in the transition probability matrix is the probability that the next state will be $j$, given that the current state is $i$. Thus, for all $i, j \in S$, we have $0 \leq P_{ij} \leq 1$, and $\sum_j P_{ij} = 1$.

We define the underlying directed graph of a Markov chain as follows: there is one vertex in the graph for each state of the Markov chain; and there is an edge directed from vertex $i$ to vertex $j$ if an only if $P_{ij} > 0$.

2.12 A stationary distribution for the Markov chain with transition matrix $P$ is a probability distribution $\pi$ such that $\pi = \pi P$. The distribution $\pi$ is also called the equilibrium distribution.

A directed graph is said to be aperiodic if there is no integer $k > 1$ that divides the length of every cycle of the graph. A Markov chain is aperiodic if only if the underlying digraph is aperiodic.

2.13 Let $G = (V, E)$ be a connected digraph where $|V| = n$ and $|E| = m$. From $G$ we construct a Markov chain $M_G$. The states of $M_G$ are the vertices of $G$, and for any two vertices $u, v \in V$, $P_{uv} = \begin{cases} \frac{1}{d^+(u)} & \text{if } (u, v) \in E \\ 0 & \text{otherwise,} \end{cases}$

where $d^+(u)$ is the out-degree of vertex $u$. $M_G$ can viewed as a simple random walk on $G$.

Theorem 2.1: If $G$ is a strongly connected aperiodic digraph and $M_G$ is the induced Markov chain, then $M_G$ has a unique stationary distribution $\pi$.

2.14 The hitting time $\text{Hit}(u, v)$ of a digraph $G$ is the expected number of steps for a random walk starting from $u$ to reach $v$. The hitting time of digraph $G$, $\text{Hit}(G)$, is the maximum of $\text{Hit}(u, v)$ over all pairs of vertices of $G$. 

11
2.15 The **commute time** between $u$ and $v$, $\text{Commut}(u, v) = \text{Hit}(u, v) + \text{Hit}(v, u)$.

2.16 The **cover time**, $\text{Cover}(u)$, is the expected number of steps for a random walk on $G$ starting from $u$ to visit all vertices. The cover time of $G$, $\text{Cover}(G)$, is the maximum of $\text{Cover}(u)$ over all vertices $u$ of $G$.

The **mixing time** of a Markov chain is the time when the Markov chain becomes "close" to its stationary distribution. There are several measures for distance between distributions. For example, if we fix the distance metric as the Chebyshev distance and the closeness parameter as $1/4$ then the mixing time can then be defined as $t_{\text{mix}}(\varepsilon) = \min\{t : d(t) \leq 1/4\}$, where $d(t) = \max_{v \in V}\{|\mu P^t(v) - \pi(v)|\}$, and $\mu$ is the initial distribution.

The **undirected s-t connectivity (USTCON) problem** is the following: given an undirected graph $G$ and two vertices $s$ and $t$ in $G$, decide whether $s$ and $t$ are in the same connected component. The same problem for digraphs is called **s-t connectivity (STCON) problem**. USTCON is a well-known complete problem for $SL$, while STCON is a complete problem for $NL$. 

Chapter 3

Relevant Prior Work

There are some well-known bounds on cover time on undirected graphs: \( n^2 \) for Line graphs, \( n \log n \) for expender graphs, and \( n^3 \) for lollypop graphs [Feige]. For directed graphs, the cover time can be exponentially large. An analysis of the convergence and cover times for random walks on digraphs is really an analysis of the “flow” of probability through the network. In this sense the electrical analogy can be applied. It is well known from the theory of network flows that the Min-cut, the smallest bottleneck in an undirected graph, gives a bound on flow through the network. The conductance of a cut is the ratio of the edges crossing the cut and the smaller of the two sets of vertices comprising the cut. The graph conductance, \( h(G) \), is the minimum conductance over all cuts. The study of the properties of undirected graphs in relation to the eigenvalues associated with the Laplacian matrix seems yields rich results, and can be applied directly to the behavior of random walks, yielding bounds on mixing and cover time, and relating the Laplacian eigenvalues to the conductance.

3.1 Electric Networks and Cover Time

The relation between random walks on undirected graphs and electrical networks has been well studied. Doyle and Snell [DS] stated the basic results showing the correspondence between random walks on connected undirected graphs and electrical networks. In this section we survey these results
We transform a given graph $G$ to an electrical network by placing a unit resistance $r_{uv}$ on each edge. (see Figure 3.1) The conductance of the edge $c_{uv}$ equals to $1/r_{uv}$. Let the conductance at $u c_u = \Sigma_v c_{uv}$ for all neighbors $v$ of $u$, and let transition probability $p_{uv} = c_{uv}/c_u$, where $p_{uv}$ is the probability of taking edge $(u, v)$ at vertex $u$.

Figure 3.1: Graph as an electrical network

The effective resistance $R(u, v)$ between two vertices $u$ and $v$ is the potential difference required to send one unit of current from $u$ to $v$ and let $\phi(u, v)$.

**Lemma 3.1:** $o(1) \leq R(u, v) \leq n - 1$

Chandra et al. [5] showed that the length of a random walk on a graph is related to the resistance in the corresponding electrical network. To summarize:

**Lemma 3.2:** Let $\phi(u, v)$ donate the potential difference between $v$ and any other node $u$, and let $d(x)$ donate the degree of $x$. If $d(x)$ units of current are injected into each node $x \in V$, and $2m = \Sigma_x d(x)$ units of current are removed from $v$, then

$$Hit(u, v) = \phi(u, v) \quad \forall u \in V.$$

**Lemma 3.3:** $d(u) = \Sigma_{w \in N(u)} (\phi(u, v) - \phi(w, v)) \quad \forall u \in V - \{v\}.

**Lemma 3.4:** $Hit(u, v) = \Sigma_{w \in N(u)} 1/d(u)(1 + Hit(w, v)) \quad \forall u \in V - \{v\}.$
If we identify $\phi(u, v)$ with $Hit(u, v)$ lemma 3.3 and 3.4 show that they are identical. We then obtain:

**Theorem 3.5:** For a given connected undirected graph $G$ with $m$ edges, and any two vertices $u$ and $v$

$$Commute(u, v) = 2mR(u, v).$$

**Lemma 3.6:** For a given connected undirected graph $G$ with $m$ edges, if $Commute(u, v)$ is equal to $k$, then adding an edge to $G$, increases $Commute(u, v)$ by less than $(1/m)k$.

The electrical resistance of a graph $R(G)$ is the maximum effective resistance between any pair of vertices.

**Theorem 3.7:** For a given connected undirected graph $G$ with $n$ vertices and $m$ edges,

$$mR(G) \leq Cover(G) \leq O(mR(G)\log n).$$

**Theorem 3.8:** Given a connected undirected graph $G$, let $G'$ be a complete graph derived from $G$, and let the weight of edge $(u', v')$ equal $R(u, v)$. Let $R(span)$ be the weight (or equivalently, total resistance) of the minimum spanning tree in $G'$.

$$Cover(G) \leq 2mR(span).$$

**Lemma 3.9:** If minimum degree of a undirected graph $G$ is $d$, then $R(G) \geq 1/d$.

**Lemma 3.10:** If a undirected graph $G$ contains $p$ edge-disjoint paths of length less than or equal to $l$ from $u$ to $v$, then $R(u, v) \leq l/p$.

**Theorem 3.11:** For any $n$-vertex, $d$-regular undirected graph $G$ with $d \geq \lceil n/2 \rceil$, $R(G) \leq 4/d = O(1/n)$. Hence $Cover(G) = O(n \log n)$.

Given a connected undirected graph $G$ with $n$ vertices, let $D$ be the diagonal matrix, and let $A$ be the adjacency matrix of $G$, and define the Laplacian matrix $L = D - A$. Define $\sigma(G)$ be the second smallest eigenvalue of $L$, and it is also called the spectral gap. $1-\lambda(G)$ gives the
expansion of the graph, where $\lambda(G)$ is the 2\textsuperscript{nd} largest eigenvalue of $A$.

**Theorem 3.12:** If $G$ is an undirected connected graph on $n$ vertices, with minimum degree $d_{\text{min}}$ and maximum degree $d_{\text{max}}$, then

$$1/n\sigma(G) \leq R(G) \leq 2/\sigma(G),$$

and

$$(1 - \lambda(G))d_{\text{min}} \leq \sigma(G) \leq (1 - \lambda(G))d_{\text{max}}.$$

**Lemma 3.13:** If $G$ is an $(n, d, \lambda)$-expander, then $\sigma(G) \geq \lambda^2/(4 + 2\lambda^2)$ by Alon(1986), hence $R(G) \leq (2 + \lambda^2)/\lambda^2$.

**Theorem 3.14:** A connected $(n, d, \lambda)$-expander $G$ has resistance at most $24/(\lambda^2(d + 1))$.

### 3.2 Circulation and directed Cheeger Constants

The situation changes in directed graphs since cuts have directed edges crossing them and the flow of probability is dependent on these directions. However the notion of conductance (or the Cheeger constant) can be recovered in the directed case, and relates directly to cover time and mixing time ([Fill] and [Chung]). Since the adjacency matrices for digraphs are in general asymmetric (and the corresponding Markov chains irreversible), the graph Laplacian as defined for undirected graph is no longer meaningful. However, some modification of these methods, such as a digraph Laplacian matrix and the circulation on a directed graph, can be applied to those digraphs which have some symmetry and balance properties that make them sufficiently similar to undirected graphs.

**Lemma 3.15:** If a digraph $G$ is strongly connected and aperiodic, the random walk converges to a unique stationary distribution $\pi$. 
**Lemma 3.16:** For an undirected graph, \( \pi(v) = d(v)/\sum_u d(u) = d(v)/2m \) where \( m \) is the number of edges.

**Lemma 3.17:** A digraph is balanced, if for each vertex, its in-degree and out-degree are the same. For a connected and balanced digraph on \( n \) vertices, \( \pi(v) = d(v)/\sum_u d(u) = d(v)/m \).

**Lemma 3.18:** A digraph is regular if for all vertices, their in-degree and out-degree are the same. For a regular digraph on \( n \) vertices, \( \pi = 1/n \).

**Lemma 3.19:** For a strongly connected digraph \( G \) on \( n \) vertices, \( \max \pi(v) \leq kD \min \pi(y) \), where \( D \) denotes the diameter of \( G \) and \( k \) denotes the maximum out-degree.

For a directed graph \( G \), let a function \( F: E(G) \to \mathbb{R}^+ \cup \{0\} \) assign to each directed edge \((u, v)\) a non-negative value \( F(u, v) \). \( F \) is a **circulation** if at each vertex \( v \), \( \sum_u F(u, v) = \sum_w F(v, w) \) for \( \forall \) edge \((u, v)\), edge \((v, w)\) \( E(G) \).

**Lemma 3.20:** For a directed graph \( G \), the eigenvector \( \phi \) of the transition probability matrix \( P \) having eigenvalue 1 is associated with a circulation \( F_\phi \) as follows: for each directed edge \((u, v)\) \( E(G) \), \( F_\phi(u, v) = \phi(u)P(u, v) \).

A circulation is reversible if \( F(u, v) = F(v, u) \). For a circulation \( F_\phi \), the flow at a vertex \( v \) is given by \( \phi(v) = \sum_u F(u, v) = \sum_w F(v, w) \) for \( \forall \) edge \((u, v)\), edge \((v, w)\) \( E(G) \).

![Figure 3.2: An example of digraph has exponential cover time](image-url)
For undirected graphs, the flow at a vertex $u$ is proportional to its degree and is within a polynomial factor (in $n$) of the flow at any other vertex. For some directed graphs, the flows at the vertices can differ by exponential factors, as in the graph of Figure 3.2.

The out-boundary of $S$, denoted by $\partial S$, consists of all edges $(u,v)$ with $u \in S$ and $v \in \bar{S}$. We write $F(\partial S) = \sum_{u \in S, v \in \bar{S}} F(u, v)$. If $F$ is a circulation, then $F(\partial S) = F(\partial \bar{S})$ where $\bar{S}$ denotes the complement of $S$. For a vertex $v$, $F(v) = \sum_u F(u, v)$ for all edges $(u, v) \in E(G)$ and $F(S) = \sum_v F(v)$ for all $v \in S$. For a strongly connected graph $G$ with stationary distribution $\phi$, we consider the circulation flow $F_\phi$ and define the Cheeger constant (also called the digraph conductance) as follows:

$$h(G) = \inf_S \frac{F_\phi(\partial S)}{\min\{F_\phi(S), F_\phi(\bar{S})\}}$$

where $S$ ranges over all non-empty proper subset of the vertex set of $G$.

**Theorem 3.21:** Let $G$ be a directed graph with eigenvalues $\sigma_i$ of the Laplacian. Then $\sigma(G)$, the second smallest eigenvalue, satisfies $2h(G) \geq \sigma(G) \geq h^2(G)/2$ where $h(G)$ is called the Cheeger constant or edge expansion of $G$.

**Lemma 3.22:**

1. For a strongly connected regular digraph $G$ on $n$ vertices and degree $k$, $h(G) \geq 2/kn$.
2. For a strongly connected balanced digraph $G$ on $m$ edges, $h(G) \geq 2/m$.
3. For some directed graphs $G$ with bounded out-degrees, the Cheeger constant of $G$ can be exponentially small, i.e., $h(G) \leq c^{-n}$ for some constant $c$.

**Theorem 3.23:** Some strongly connected directed graphs have Laplacian eigenvalues that are exponentially small (in $n$).
We remark that for a connected undirected graph on \( n \) vertices, the eigenvalue \( \sigma(G) \) of the Laplacian is at least \( 1/n^2 \). Convergence is often measured in either the so-called total variation distance or the relative pointwise distance. A lazy walk is a random walk that at each time step chooses to stay at current vertex with probability \( 1/2 \).

**Theorem 3.24:** A strongly connected directed graph \( G \) on \( n \) vertices has a lazy random walk with convergence rate \( 2\sigma(G)^{-1}(\log \min_u \phi(u)) \).

**Theorem 3.25:** A strongly connected regular directed graph \( G \) on \( n \) vertices with out-degree \( k \) has a lazy random walk converging to the uniform distribution with the rate of convergence no more than \( k^2n^2\log n \).

**Theorem 3.26:** A strongly connected balanced digraph \( G \) with \( m \) edges has a lazy random walk with the rate of convergence no more than \( m^2\log m \).

### 3.3 SL = L and Promise-RL

The main idea behind Reingold’s algorithm is essentially this: Normally to search an graph one needs to maintain a stack of vertices, which can grow to height \( n \), and thus use \( \Theta(n \log n) \) space. We can represent the graph by a rotation map instead of an adjacency matrix. This allows us to do a depth first search by stacking the edges followed, rather than vertices. Reingold showed how to transform any undirected graph (using log space) into an expander with constant degree. The key property of expansion is used to insure that if there is a path in the original graph between two nodes, then there is a path in the transformed graph, between the corresponding two nodes, of length at most \( \Theta(\log n) \). Since the transformed graph has constant degree, we could search the graph using a constant number of bits for each edge stacked, and the stack would never grow to more than \( \Theta(\log n) \) height. We can actually perform the search without a stack
since we can systematically enumerate all paths of length $O(\log n)$, as edge sequences, using $O(\log n)$ space. Additionally, the search algorithm never actually writes out the transformed graph, but produces portions of it on demand, by a clever recursion that is bounded by log space.

The idea behind constructing the expander, i.e., bounding the path length, is to use powering. For example if we raise a graph to the $l^{th}$ power (using either an adjacency matrix or a rotation map), we get the connectivity graph, in which two vertices are adjacent iff there is a path of length $\leq l$ between those two vertices in the original graph. However the degree of the graph will explode. Reingold found a way to lower the degree of a graph (the zig-zag product) without harming its expansion properties (equivalently, path length properties). The transformation involves taking a power of the graph, followed by performing a zig-zag product to lower the degree. This process is iterated until the desired expansion (path length) property is achieved.

3.27 For a given undirected graph $G(V, E)$ and two vertices $s, t \in V$, the algorithm will return “yes” if and only if there exists a path in $G$ between $s$ and $t$.

**Step 1:** Convert $G$ into a regular graph $G_{reg}$.

First, we replace each vertex $v$ of $G$ of degree $d_v$ by a cycle of length $d_v$, and then place the edges of $G$ as a matching in the new graph. This new graph is polynomial in the size of the original and reduces the degree to 3. We then add $(d^{16} - 3)$ self-loops to each vertex to yield $G_{reg}$, which is now a $d^{16}$-regular, non-bipartite graph. The purpose of increasing the degree to $d^{16}$ is because we will take the zig-zag product of the graph with an expander and for this operation we require an expander with size matching the degree of the regular graph. It is known how to explicitly construct a degree $d$ expander of size $d^{16}$.
3.28 For a $D$-regular undirected graph $D$, the rotation map $\text{Rot}_G : [N] \times [D] \rightarrow [N] \times [D]$ is defined as follows: $\text{Rot}_G(v, i) = (w, j)$ if the $i^{\text{th}}$ edge incident to $v$ leads to $w$, and this edge is the $j^{\text{th}}$ edge incident to $w$.

It is well known that the expansion of a graph is related to the spectral properties of its adjacency matrix. $1 - \lambda(G)$ gives the expansion of the graph, where $\lambda(G)$ is the 2nd largest eigenvalue. So to retain large expansion we must keep $\lambda(G)$ small.

Lemma 3.29: For every $D$-regular, connected, non-bipartite graph $G$ on $[N]$ it holds that $\lambda(G) \leq 1 - 1/DN^2$.

Step 2: Converting $G_{\text{reg}}$ to an expander $G_{\text{exp}}$.

Let $H$ be a $d$-expander with $d^{16}$ vertices. Applying powering on $G_{\text{reg}}$ improves the expansion. Applying the zig-zag product $G \odot H$ reduces the degree without harming the expansion. Combine these two methods to define a family of graphs as follows:

$$G_0 = G_{\text{reg}}$$

$$G_i = (G_{i-1} \odot H)^8$$
Let $i = \log d^{16} n^2 = O(\log n)$, be the smallest integer such that $(1 - \frac{1}{d^{16} n^2})^{2^i} \leq \frac{1}{2}$, then the number of vertices of $G_i$ is $n(d^{16})^i$, which is polynomial in $n$ and $\lambda(G_i) \leq 1/2$.

3.30 Let $G$ be a $D$-regular multi-graph on $[N]$ given by rotation map $\text{Rot}_G$. The $t^{th}$ power of $G$ is the $D^t$-regular graph $G^t$ whose rotation map is given by $\text{Rot}_G^t(v_0, (a_1, \ldots, a_t)) = (v_t, (b_t, \ldots, b_1))$, where these values are computed via the rule $(v_i, b_i) = \text{Rot}_G(v_{i-1}, a_i)$.

3.31 If $G$ is a $d$-regular graph on $[N]$ with rotation map $\text{Rot}_G$ and $H$ is a $d$-regular graph on $[D]$ with rotation map $\text{Rot}_H$, then their zig-zag product $G \circ z H$ is defined to be the $d^2$-regular graph on $[N] \times [D]$ whose rotation map $\text{Rot}_{G \circ z H}$ is as follows:

$\text{Rot}_{G \circ z H}((v, a), (i, j))$:
1. Let $(a', i') = \text{Rot}_H(a, i)$.
2. Let $(w, b') = \text{Rot}_G(v, a')$.
3. Let $(b, j') = \text{Rot}_H(b', j)$.
4. Output $((w, b), (j', i'))$.

Figure 3.4: An example of zig-zag product

Lemma 3.32: Let $G^t$ is the $t^{th}$ power of $D$-regular graph $G$, then $G^t$ is a $D^t$-regular graph
Lemma 3.33: If $\lambda(H) \leq \frac{1}{2}$, then $1 - \lambda(G \oplus H) \geq \frac{3}{8}(1 - \lambda(G))$.

Step 3: Check if $s$ and $t$ are connected in $G_{exp}$ by enumerating all $O(\log n)$ length paths originating at $s$.

Lemma 3.34: For every expander graph $E$ and any two vertices $s$ and $t$ in $E$, there exists a path of length $O(\log n)$ that connects $s$ to $t$.

The work of transforming the input graph $G$ into an expander graph is represented by the rotation map and done by recursive calls to construct intermediate graphs, which ultimately use the original adjacency matrix as the base case. This matrix appears on the input tape. We need to bound the space used by the recursion stack. The height of the stack is at most $O(\log n)$. Each squaring operation requires an additional space of $O(\log \deg(G))$ and the zig-zag product with $H$ requires additional space of at most $O(\log \deg(G))$. Since the degree of each graph in the family is at most $d^{16}$, which is a constant, the entire recursion stack uses $O(\log n)$ space. The counter for enumerating all possible paths requires $O(\log n)$ space. Thus, Reingold’s algorithm is a log-space deterministic algorithm for USTCON.

Research on extending Reingold’s algorithm to the directed case has been undertaken by Chung, Reingold and Vadhan (2011), in an attempt to prove $RL = L$. Their best result shows that STCON on a poly-mixing digraph with a known stationary distribution is in $L$. They show that STCON on a poly-mixing digraph is promise-$RL$ complete. The complexity of searching such a digraph, without knowing the stationary distribution, remains open.

3.35 SHORT-WALK S-T CONNECTIVITY:

- Input: $(G, s, t, 1^k)$, where $G = (V, E)$ is a directed graph, $s, t \in V$. 

• YES instances: A random walk of length $k$ started from $s$ ends at $t$ with probability at least $\frac{1}{2}$.

• NO instances: There is no path from $s$ to $t$ in $G$.

3.36 POLY-MIXING S-T CONNECTIVITY:

• Input: $(G, s, t, 1^k)$, where $G = (V, E)$ is a out-regular directed graph, $s, t \in V$, and $k \in \mathbb{N}$.

• YES instances: The random walk on $G$ has a stationary distribution $\pi$ such that $\lambda_{st}(G) \leq 1 - \frac{1}{k},$ and $\pi(s), \pi(t) \geq \frac{1}{k}$.

• NO instances: There is no path from $s$ to $t$ in $G$.

3.37 POLY-MIXING FIND PATH:

• Input: $(G, s, t, 1^k)$, where $G = (V, E)$ is a out-regular directed graph, $s, t \in V$, and $k \in \mathbb{N}$.

• Promise: $\lambda_{st}(G) \leq 1 - \frac{1}{k},$ and $\pi_s(s), \pi_t(t) \geq \frac{1}{k}$.

• Output: A path from $s$ to $t$ in $G$.

**Theorem 3.38:** POLY-MIXING S-T CONNECTIVITY is complete for promise-RL. POLY-MIXING FIND PATH is complete for search-RL.

### 3.4 ReachFewL = ReachUL

*FewL = UL* is an interesting open question and a solution is likely to have implications on the NL versus UL question. Recent research gives ample evidence to believe that the conjecture NL = UL is true. However, researchers have yet to find a proof of this equality. Lange showed that the directed graph reachability problem associated with reach-unambiguous computations is *ReachUL-complete* [Lange]. Allender and Lange showed that this reachability problem can be
solved deterministically in space $O(\log^2 n / \log \log n)$ which is asymptotically better than the Savitch’s $O(\log^2 n)$ bound for the general reachability problem [AL]. $ReachUL$ is also known to be closed under complement.

Let $G$ be a digraph, $s$ be a vertex in $G$ and $k$ be an integer. We say that $G$ is $k$-reach-unambiguous with respect to $s$ if for all vertices $v \in V$, there are at most $k$ paths from $s$ to $v$. If $k = 1$, we say $G$ is reach-unambiguous with respect to $s$.

**Theorem 3.39:** $ReachFewL = ReachUL$.

Let $G$ be a digraph on $n$ vertices and $s$ be a vertex of $G$. We say that $G$ is distance isolated with respect to $s$, if for every vertex $v \in V$ and weight $w \in \{1, \ldots, n\}$, there is at most one path of weight $w$ from $s$ to $v$.

**Lemma 3.40:** Let $E = \{e_1, e_2, \ldots, e_m\}$ be the set of edges of $G$. Let $q$ be a polynomial. If $G$ is $q(n)$-reach-unambiguous with respect to $s$, then there is a prime $p \leq n^k$, for some constant $k$, such that the weight function $w_p : E \rightarrow \{1, \ldots, p\}$ given by $w_p(e_i) = 2^{i} \pmod{p}$ defines a weighted graph $G_{wp}$ which is distance isolated with respect to $s$.

The layered graph $lay(G)$ induced by $G$ is the graph on vertices $V \times \{0, 1, \ldots, n\}$ and for all edges $(u, v)$ of $G$ and $i \in \{0, 1, \ldots, n - 1\}$, the edge $(u, i) \rightarrow (v, i + 1)$ is in $lay(G)$.

**Lemma 3.41:** If $G$ is an acyclic and distance isolated graph with respect to a vertex $s$, then $lay(G)$ is reach-unambiguous with respect to $(s, 0)$, and there is a path of length $d$ from $s$ to $v$ in $G$ if and only if there is a path from $(s, 0)$ to $(v, d)$ in $lay(G)$.

Let $lay(G_{wp})$ be a layer digraph obtained from $G$ followed the constructions of lemma 3.40 and 3.41. $lay(G_{wp})$ is reach-unambiguous with respect to $(s, 0)$. Moreover, there is a path from $s$ to $t$ in $G$, if and only if there is a $d$ such that there is a path from $(s, 0)$ to $(t, d)$. 


The reachability problem for 3-page graphs is complete for NL, for 2-page graphs is equivalent to reachability in grid graphs and hence is in UL, and for 1-page graphs (which are outer-planar graphs) is complete for L.

3.42 ThreePage is the class of all graphs $G$ that can be embedded on 3 pages as follows: all vertices of $G$ lie along the spine and the edges lie on exactly one of the three pages without intersection. Moreover all edges are directed from top to bottom.

3.43 THREEPAGEREACH is the language consisting of tuples $(G, s, t, f)$, such that $G \in$ ThreePage, $s$ and $t$ are two vertices in $G$ and there exists a path from $s$ to $t$ in $G$, and $f$ is an embedding of $G$ on 3 pages (that is, $f$ defines the ordering of the vertices along the spine and in which page an edge lies on).

Theorem 3.44: THREEPAGEREACH is complete for NL.

![Figure 3.5](image)

(a) Graph $G$ (b) The corresponding 3-page graph $H$. The dashed edges are on the page 3.

Let $H$ be a 3-page graph obtained from a given digraph $G$. Label the vertices of $G$ with \{1, 2, ..., $n$\} in a topological order and label the edges of $G$ with \{1, 2, ..., $m$\}. $H$ contains $2mn$ vertices: for each vertex $u_i$ on $G$, there are $2m$ corresponding vertices of $H$ and they are ordered by \{11, 21, ..., $n1$, $n2$, $(n-1)2$, ..., $1(2m-1)$, $2(2m-1)$, ..., $n(2m-1)$, $n(2m)$, $(n-1)(2m)$, ..., $1(2m)$\}. There is an edge $(u_{ij}, u_{(i+1)j})$ for $1 \leq i \leq n$ and $1 \leq j < m$. For each edge $e_i (x, y)$ of $G$, there
is an edge \((x_{2i-1}, y_{2i})\) of \(H\). Thus, there is a path from \(x\) to \(y\) in \(G\), if and only if there is a path from \(x_i\) to \(y_j\) in \(H\).

In a different direction, we are interested in the complexity of the NL class. If NL is not equal to L, and if RL = L, as is generally believed, then RL is a small subset of NL and there is a large gap between RL and NL. We aim to categorize the problem of STCON on classes of strongly connected digraphs which will correspond to different (potential) sub-NL complexity classes. We will adapt the techniques and methods which have been developed in the study of USTCON for use on these new classes.

Another measure similar to the conductance is the spectral expansion of a digraph [Calude]. The weakness of all these measures is that they bound the mixing time but not the cover time. In particular, one can have rapidly mixing digraphs (polynomial convergence time) which nonetheless have exponential cover time. To remedy this situation we introduced a new, basic measure of the imbalance of a digraph called the number of asymmetric vertices [CCZ]. This new measure yields bounds on the cover time. Asymmetric vertices are the vertices of a digraph whose out-edges are not in a maximal balanced subgraph. Our analysis of the measure proceeds by first studying the effect of adding edges to an initially balanced digraph. We then use this result to obtain the stated bounds as a function of the number of asymmetric vertices.
Chapter 4

Weakly Symmetric Classes

We now motivate our decision to consider only strongly connected digraphs with the following discussion. We note that there is a simple construction that reduces RL (or NL) to a search on a single strongly connected digraph component (see [CRV]), but we wish to retain the property that each node of $G$ has bounded degree. We call an order on the vertices of a digraph *pseudo-topological* if the removal of the edges to the least vertex makes it a topological order. We observe that every NL computation can be simulated by a “weakly symmetric” machine $M$, in the following manner. We may assume w.l.o.g. that all computation paths lead to terminal configurations and that there is a unique accepting configuration. From each non-accepting terminal configuration of the original machine, the simulating machine $M$ erases the work tapes, rewinds the input tape and returns to the start configuration $S$. Given an upper bound $b$ on the number of distinct configurations of $M$ on input $x$, we can construct a strongly connected computation graph for $M$ with a pseudo-topological order. The nodes will consist of the pair $(0, S)$, where $S$ is the start configuration and the pair $T = (b, A)$, where $A$ is the accepting configuration, together with all pairs $(t, C)$, where $0 < t \leq b$ is an integer time step and $C \neq S$ is a configuration. There is an edge from a node $(t, C_i)$ to $(t+1, C_j)$ iff there is a valid transition of $M$ from configuration $C_i$ to $C_j$ or $C_i = C_j = A$, and an edge from $(t, C_i)$ to $(0, S)$ iff there is a transition from $C_i$ to $S$. Since we may assume a bound on the number of nondeterministic choices machines can make (typically two) the resulting computation graph has out-degree bounded by
2, and the subgraph consisting of the nodes reachable from \((0, S)\) has in-degree bounded by a constant, depending on the machine and the alphabet size. The natural order on these pairs becomes a topological order on the nodes if the edges to \((0, S)\) are removed, and thus it is pseudo-topological.

### 4.1 Strong Chains

Any successful computation path can be represented as a directed path in a graph. For this initial study of computation graphs we will focus on these paths. Following all computation paths of a weakly symmetric machine will yield a single Strong Chain, but one with exponential size and redundancy. The actual computation graph, without this redundancy, will be a union of Strong Chains, and have polynomial size. The study of Strong Chains provides a good tool for the analysis of strongly connected digraphs, and we are able to transfer our analysis to general digraphs.

The vertex \(v_0\) is the initial state, and the vertex \(v_n\) is the successful terminal state.

A **Strong Chain** is a simple strongly connected directed graph which contains a directed Hamiltonian path as a subgraph.

A random walk is a stochastic (Markov) process in which each step starts from the current vertex (state) and selects an out-neighbor of that vertex, uniformly at random, to visit and thus become the new current vertex. Our first goal is to estimate the hitting time from \(v_0\) to \(v_n\) for a given digraph. For this purpose we define some terminology for Strong Chains.
4.1.1 Strong Chain \((V, F, B)\)

1. A set \(V\) of vertices, labeled from 0 to \(n\).
2. The directed Hamiltonian path of length \(n\), called base, consists of directed edges from \(v_i\) to \(v_{i+1}\) for \(0 \leq i < n\).
3. The edge distance of an edge from \(v_i\) to \(v_j\) is \(|i - j|\).
4. A forward edge is an edge from \(v_i\) to \(v_j\) where \(i < j\).
5. The set \(F\) of directed forward edges (e.g. edges a and b in Figure 4.1) satisfies \(|F| \geq n\). The forward edges of distance 1 are called “to next” (edge a).
6. \(F_0\) is the base, the set of the edges of “to next”, and \(|F_0| = n\).
7. A back edge is an edge from \(v_i\) to \(v_j\) where \(i > j\).
8. A set \(B\) of directed back edges (edges c and d). The back edges of distance 1 are called “back to previous” (edge c).
9. A back edge from \(v_i\) to \(v_0\) (edge d) is called “back to root”.
10. \(\text{Hit}(i, j)\) is the expected number of steps taken by a random walk on a Strong Chain starting from \(v_i\) and reaching \(v_j\).
11. The hitting time of a Strong Chain $C$, $\text{Hit}(C) \geq \text{Hit}(0, n)$.

12. The cover time a Strong Chain $C$, $\text{Cover}(C)$, is the expected time for a random walk started from $v_0$ to visit all vertices.

### 4.2 Strong Chain Symmetry

For a directed graph $G(V, E)$,

1. A directed edge is *symmetric* if its inverse edge also belongs to $G$. That is, edge $(u, v)$ and edge $(v, u) \in E$.

2. A vertex is *symmetric* if all of its edges are symmetric.

3. A vertex is *balanced* if its out-degree is equal to its in-degree.

4. $G$ is *symmetric* if all of its edges are symmetric. A directed symmetric graph is equivalent to the corresponding undirected graph.

5. $G$ is *balanced* if all of its vertices are balanced, and is called a *pseudo-symmetric* digraph. A connected balanced digraph is called an *Eulerian Circuit* digraph, and it contains a directed Eulerian circuit.

6. $G$ is *regular* if the in-degree and out-degree of all vertices are equal. A regular digraph has *symmetry* (but is not a symmetric digraph in the sense defined above).

We note that using these definitions for connected digraphs, *symmetric* implies *balanced*, *balanced* implies *Eulerian Circuit*, and *regular* implies *balanced* and *Eulerian Circuit*. 
4.3 Symmetric and Asymmetric Graphs

We begin our analysis of Strong Chains by noting the effect of adding back edges to the base. This analysis will be important in demonstrating the domination property of Harps in the next section, and in illustrating the reason for our choice of symmetry property. We then perform a similar analysis for the Line digraph, defined below. We then transfer this analysis to Strong Chains.

![Harp Strings Digraph](image)

Figure 4.2: Harp Strings Digraph

The following simple Lemma will be useful.

**Lemma 4.1:** For a given Strong Chain $C$,

$$\text{Hit}(0,n) \leq \text{Cover}(C) \leq \sum_{i=0}^{n-1} \text{Hit}(i, i + 1).$$

Equality is only satisfied when the forward edges satisfy $F = F_0$.

The digraph in Figure 4.2 we will call the *Harp Strings* digraph, in anticipation of our definition of the *Harp* digraph in the next section. In addition to the base, each vertex $v_i$, $i > 0$, has a “back to root” edge.

**Lemma 4.2:** For the *Harp Strings* digraph $G$, $\text{Hit}(0,n) = \text{Cover}(G) = \Theta(2^n)$.

**Proof:** The probability of advancing from $v_i$ to $v_{i+1}$ and the probability of returning to the start are both $\frac{1}{2}$. $\text{Hit}(0,n)$ is thus the expected time of a run of $n$ successes in a Bernoulli process, which by solving the well-known recurrence is $\Theta(2^n)$. $\square$
The Line graph consists of a single simple path from $s$ to $t$ and has diameter $n-1$. The Line digraph, Figure 4.3, is obtained from the Line graph by replacing each undirected edge with two directed edges, and thus also has diameter $n-1$.

**Lemma 4.3**: For the Line digraph $G$, $\text{Hit}(0, n) = \text{Cover}(G) = n^2$.

**Proof**: This is also a well-known simple recurrence.

$\text{Hit}(0, 1) = 1$

$\text{Hit}(i, i + 1) = \frac{1}{2} \times 1 + \frac{1}{2} (1 + \text{Hit}(i - 1, i) + \text{Hit}(i, i + 1)), \text{for } 1 \leq i < n$

$\text{Hit}(i, i + 1) = 2 + \text{Hit}(i - 1, i) = 2i + 1$

From Lemma 4.1, we have $\text{Hit}(0, n) = \sum_{i=0}^{n-1} \text{Hit}(i, i + 1)$ where $F = F_0$ then

$\text{Hit}(0, n) = \sum_{i=0}^{n-1}(2i + 1) = n^2$ 

It is not necessary to have all back edges *back to root*, or a badly unbalanced digraph to have exponential hitting time. A connected directed graph has an Eulerian path if and only if at most one vertex has out-degree − in-degree = 1, at most one vertex has in-degree − out-degree = 1 and every other vertex is balanced. If we modify the Line digraph by changing each back edge to be distance 2 (see Figure 4) then we obtain a Strong Chain $C_E$, which is an *Eulerian path* digraph with bad behavior.
Lemma 4.4: For $C_E$, $Hit(0, n) = Cover(C_E) = \Theta(\varphi^n)$, where $\varphi$ is the golden ratio.

Proof:

$Hit(0, 1) = Hit(1, 2) = 1$

$Hit(i, i + 1) = \frac{1}{2} \times 1 + \frac{1}{2} \left(1 + Hit(i - 2, i - 1) + Hit(i - 1, i) + Hit(i, i + 1)\right),$

for $2 < i < n$

$Hit(i, i + 1) = 2 + Hit(i - 2, i - 1) + Hit(i - 1, i)$

From Lemma 4.1, we have $Hit(0, n) = \sum_{i=0}^{n-1} Hit(i, i + 1)$ where $F = F_0$. $Hit(i, i + 1)$ grows in proportion to the well known Fibonacci sequence, so that $Hit(0, n)$ grows as the sum of the first $n$ Fibonacci numbers, yielding: $Hit(0, n) = \Theta(\varphi^{n+2}) = \Theta(\varphi^n)$. □

Notice that although $C_E$ has only two unbalanced vertices, half of the back edges are not included in any maximal balanced subgraph. This observation is the basis of our symmetry measure. $C_E$ also illustrates that some badly behaved digraphs can be easily balanced in a way that removes the bad behavior. If we add a single long forward edge of length $n-2$ on the two unbalanced vertices to $C_E$ we obtain an Eulerian Circuit digraph $C_E'$ with both polynomial hitting and cover time.
Lemma 4.5: For $C_E'$ the hitting and cover time is polynomial in $n$.

Proof: A bound on the cover time follows directly from a result of [Chung], which states that a strongly connected Eulerian directed graph $G$ with $m$ edges has a lazy random walk with the rate of convergence no more than $m^2 \log m$. The hitting time is thus also bounded by a polynomial. □
In Figure 4.6 we see an example of a 2-regular digraph. We mention that Chung has demonstrated that the rate of convergence for a $k$-regular strongly connected digraph is no more than $k^2 n^2 \log n$.

Next we compare several ways of adding the same number of asymmetric directed edges to the Line digraph. If we add a back to root edge to each vertex, except for $v_0$, $v_1$, and $v_n$, we obtain a graph $L_1$ (see Figure 4.7) with exponential hitting time. The analysis is similar to Harp Strings digraph. We remark that the hitting time of any Strong Chain (and any digraph in general) with maximal out degree $d$ will be bounded by $O(d^n)$. This is because the waiting time for $n$ consecutive successes in a Bernoulli experiment, with the probability of success $1/d$, is $\Theta(d^n)$ [CS]. We remark that if we add to each vertex of the base both a back to root edge and an edge to $v_1$, we will achieve this worst case behavior of $\Theta(3^n)$, for out-degree 3. If we add all possible back edges to the base we get $\Theta(n!)$ hitting time.

We summarize this discussion with the following lemmas.

**Lemma 4.6:** For any $n+1$ vertex Strong Chain $C$, with vertices $v_0, \ldots, v_n$ and with the out-degree of $v_i$ denoted by $d_i$, $Hit(C) = O(n \prod_{j=0}^{n-1} d_j)$. If, for all $i$, $d_i \leq D$, then we have that $Hit(C) = O(nD^{n-1})$, since $d_0 = 1$. If the bound $D$ is constant then in the worst case $Hit(C) = \Omega(nD^{n-1} d_j + n)$. 

![Figure 4.7: Line digraph modification $L_1$](image-url)
Proof: The proof of the upper bound is by induction on $n$, one less than the number of vertices in the Strong Chain. The worst case hitting time will be obtained when each forward edge is to next. The probability of advancing from $v_i$ to $v_{i+1}$ is $1/d_i$. For the worst case the $d_n-1$ back edges from $v_n$ go to $v_0,\ldots,v_{d_i-2}$. Since $F = F_0$, by Lemma 4.1 we have that for all $0 < i < n$, $Hit(i,n) < Hit(0,n)$. For this upper bound proof, we shall use this fact by replacing the term contributed by the $d_n-1$ back edges from $v_n$ in the recurrence by $(d_n-1)Hit(0,n)$.

Let $G_i$ be a Strong Chain with $i+1$ vertices $v_0,\ldots,v_i$ and let $Hit_i$ denote the hitting time of $G_i$.

For the base case $Hit(G_1) = 1$.

The inductive hypothesis:

$Hit(G_n) \leq n \prod_{j=0}^{n-1} d_j$

Inductive Step:

We add vertex $v_{n+1}$ to $G_n$, which adds a to next edge to $v_n$.

$Hit_{n+1}(0,n) = Hit(G_n)$.

$Hit(G_{n+1}) = Hit_{n+1}(0,n) + Hit_{n+1}(n,n+1)$ from Lemma 4.1

$Hit_{n+1}(n,n+1) \leq \frac{1}{d_n} \times 1 + \frac{d_n-1}{d_n} (1 + Hit_{n+1}(0,n) + Hit_{n+1}(n,n+1))$

$= d_n + (d_n-1)(Hit_{n+1}(0,n))$.

$Hit_{n+1}(0,n+1) = Hit_{n+1}(0,n) + Hit_{n+1}(n,n+1)$

$= Hit(G_n) + d_n + d_{n-1}Hit(G_n)$

$= Hit(G_n) + d_n + d_{n-1}Hit(G_n)$

$= d_nHit(G_n) + d_n$

$\leq d_n n(\prod_{j=0}^{n-1} d_j) + d_n$ (by the inductive hypothesis)

$= n \prod_{j=0}^{n} d_j + d_n$
The proof of the lower bound when $D$ is constant is also by induction on $n$.

The base case $Hit(G_1) = 1$.

The inductive hypothesis:

$$Hit(G_n) \geq \prod_{j=0}^{n-1} d_j + n$$

Inductive Step:

In case $d_n = 1$

$$Hit_{n+1}(n, n+1) = 1 + Hit(G_n)$$

$$\geq 1 + \prod_{j=0}^{n-1} d_j + n$$

$$= \prod_{j=0}^{n} d_j + n + 1$$

Else $d_n \geq 2$

$$Hit_{n+1}(n, n+1) = \left(\frac{1}{d_n} \times 1 + \frac{1}{d_n} \sum_{j=0}^{d_n-2} Hit_{n+1}(j, n) + Hit_{n+1}(n, n+1)\right)$$

$$= d_n + \sum_{j=0}^{d_n-2} Hit_{n+1}(j, n)$$

$$= d_n + (d_n - 1)Hit(0, n) - \sum_{j=0}^{d_n-2} Hit_{n+1}(0, j)$$

$$Hit_{n+1}(0, n+1) = Hit_{n+1}(0, n) + Hit_{n+1}(n, n+1)$$

$$= Hit(G_n) + d_n + (d_n - 1)Hit(G_n) - \sum_{j=0}^{d_n-2} Hit(G_j)$$

$$= d_n Hit(G_n) + d_n - \sum_{j=0}^{d_n-2} Hit(G_j)$$

$$\geq d_n (\prod_{j=0}^{n-1} d_j + n) + d_n - \sum_{j=0}^{d_n-2} Hit(G_j)$$

$$\geq d_n (\prod_{j=0}^{n-1} d_j + n) + d_n - D^{D-1}$$

$$\geq \prod_{j=0}^{n} d_j + d_n n - D^{D-1}$$
\[ \geq \prod_{j=0}^{n} d_j + 2n - D^{D-1} \text{ (since } d_n \geq 2) \]
\[ \geq \prod_{j=0}^{n} d_j + n + 1 \text{ (since } D \text{ is constant so that } D^{D-1} < n-1) \]

In the following let \( G_0 \) be the Line digraph.

**Lemma 4.7:** Let \( G_1 \) be obtained by adding a single back edge from \( v_i \) to \( G_0 \), then in the worst case \( \text{Hit}(G_1) = n^2 + (n - i) (i^2) = \Theta(\text{Hit}(G_0) + (n - i) (i^2)) \). For \( i = n/2 \), this is \( \Theta(n \text{ Hit}(G_0)) \).

**Proof:** For the worst case we add a back to root edge from \( v_i \). We observe that adding a back to root edge from \( v_i \) will leave \( \text{Hit}(0, i) \) unchanged.

\[ \text{Hit}_1(i, i + 1) \]
\[ = \frac{1}{3} (1 + \text{Hit}_0(0, i) + \text{Hit}_1(i, i + 1)) + \frac{1}{3} (1 + \text{Hit}_0(i - 1, i) + \text{Hit}_1(i, i + 1)) + \frac{1}{3} \]
\[ \text{Hit}_1(i, i + 1) = 1 + \text{Hit}_0(0, i) + 1 + \text{Hit}_0(i - 1, i) + 1 \]
\[ \text{Hit}_1(j, j + 1) = 1 + \text{Hit}_0(0, i) + 2 + \text{Hit}_0(j - 1, j), \text{for } i \leq j \leq n - 1 \]
\[ \text{Hit}_1(0, n) = \text{Hit}_0(0, i) + \text{Hit}_1(i, n) = \text{Hit}_0(0, n) + (n - i)(1 + \text{Hit}_0(0, i)) \]
It is easy to see that this recurrence grows most rapidly when \( i = n/c \), for constant \( c > 1 \).

Let \( i = n/2 \), then
\[ \text{Hit}_1(0, n) = n^2 + (n - i)(1 + i^2) = n^2 + \frac{n^2}{2} (1 + \frac{n^2}{4}) = \Theta(n^3) \]
Thus from Lemma 4.3,
\[ \text{Hit}_1(0, n) = \Theta(n \text{ Hit}(G_0)) = \Theta(n^3) \]

**Lemma 4.8:** Let \( G \) be a graph obtained by adding a single back edge from each of \( k \) distinct vertices of \( G_0 \), the Line digraph. Then in the worst case, we have for \( 1 \leq k < n/2 \),
\[ \text{Hit}(G) = O \left( \left( \frac{2n}{k+1} \right)^k \text{Hit}(G_0) + \text{Hit}(G_0) \right) = O \left( \left( \frac{2n}{k+1} \right)^{k+2} \right) \text{ and } \text{Hit}(G) = \Omega \left( \left( \frac{n}{k+1} \right)^k \right). \]
Proof: We obtain the worst case by adding \textit{back to root} edges to $G_0$. For a given $k$ we add \textit{back to root} edges from each of the vertices $v_{i_1}, \ldots, v_{i_k}$ of $G_0$ and let $G_k$ be the induced subgraph consisting of the vertices $v_j, j = 0, \ldots, i_k$. To simplify the analysis we will assume, w.l.o.g., that $n$ is a multiple of $k + 1$. We first show by induction on $k$ that

$$Hit(G_k) = O(Hit(G_0)) + \prod_{j=1}^{k} 2(i_j - i_{j-1}) \text{Hit}(G_0),$$

where we assume that $i_0 = 0$.

Base Case:

$$Hit(G_1) = Hit_0(0, 1) = i_1^2 \leq Hit(G_0) \leq Hit(G_0) + 2(i_1 - 0)\text{Hit}(G_0).$$

For the inductive hypothesis we assume

$$Hit(G_m) \leq Hit(G_0) + \prod_{j=1}^{m} 2(i_j - i_{j-1})\text{Hit}(G_0)$$

Inductive Step:

$$Hit_{m+1}(i_m, i_m + 1) = 1 + Hit_{m+1}(0, i_m) + 1 + Hit_{m+1}(i_m - 1, i_m)$$

$$\leq 2 + 2Hit(G_m) \quad \text{since } Hit_{m+1}(i_m - 1, i_m) \leq Hit(G_m)$$

$$Hit'(j, j + 1) = 2 + Hit(j - 1, j) \text{ for } j = i_m + 1 \text{ to } i_{m+1}$$

$$Hit(G_{m+1}) = Hit(G_m) + Hit(i_m, i_{m+1})$$

$$\leq Hit(G_m) + 2(i_{m+1} - i_m) \text{Hit}(G_m) + (i_{m+1} - i_m)^2$$

$$= \prod_{j=1}^{m+1} 2(i_j - i_{j-1})\text{Hit}(G_0) + \text{Hit}(G_0)$$

Maximizing the solution to this recurrence is equivalent to maximizing the product of the distances between the added back edges, subject to the constraint that the total distance is at most $n$. This is just the problem of finding the hyperbox of maximum volume, subject to a bound on the total side lengths, and the well known solution is the hypercube [Kazarinoff]. Thus the solution is asymptotically maximal when the distances are the same length. So to prevent adding asymmetric edges to $v_0$ or $v_n$ we choose $i_m = \frac{mn}{k+1}$. This yields $Hit(G_k) = \left(\frac{2n}{k+1}\right)^k O(Hit(G_0)) +$
\( O(\text{Hit}(G_0)) = O \left( \left( \frac{2n}{k+1} \right)^{k+2} \right) \), since the hitting time of the Line digraph \( G_0 \) is \( n^2 \). One can also prove by induction that \( \text{Hit}(G_k) = \Omega \left( \left( \frac{n}{k+1} \right)^{k} \right) \) by replacing the base case by \( \text{Hit}(G_1) \geq 1 \) and the inductive hypothesis by \( (G_m) \geq \prod_{j=1}^{m}(i_j - i_{j-1}) \).

![Figure 4.8: Line digraph modification \( L_2 \)](image)

It is easy to see that when \( k \) is constant, the constant of proportionality in the hitting time is maximized if all asymmetric edges are equally spaced in the last half of the Line digraph. However, when \( k \) is large, the effect of \( \text{Hit}(G_0) \) becomes negligible. Next we consider the effect of adding \( d \) back edges from the single vertex \( v_i \) of \( G_0 \) to obtain the digraph \( L_2 \).

**Lemma 4.9:** In the worst case, \( L_2 \) has \( \text{Hit}(0, n) = \Theta(n(d \text{ Hit}(G_0))) = \Theta(n(dn^2)) \).

**Proof:** We analyze the worst case by adding long edges. We add back edges from \( v_i \) to each of the vertices \( v_0, \ldots, v_{d-1} \).

\[
\text{Hit}(i, i + 1) = \frac{1}{d+1} \sum_{j=0}^{d-1} (1 + \text{Hit}(j, i) + \text{Hit}(i, i + 1)) + \frac{1}{d+1} \\
= d + \sum_{j=0}^{d-1} \text{Hit}(j, i) \\
= d + \sum_{j=0}^{d-1} (\text{Hit}(0, i) - \text{Hit}(0, j))
\]
Let \(d = n-2\), so that \(i = n-1\)

\[
Hit(0, n) = (n-1)^2 + n - 2 + \sum_{j=0}^{n-2} (Hit(0, n - 1) - Hit(0, j))
\]

\[
= (n-1)^2 + n - 2 + (n-2)(n-1)^2 - \sum_{j=0}^{n-3} j^2
\]

\[
= (n-1)^2 + n - 2 + (n-2)(n-1)^2 - 1/6(n-3)(n-2)(2n-5)
\]

\[
= \Theta(n^3)
\]

Let \(d = n/2 - 1\), and \(i = n/2\)

\[
Hit(0, n) = \frac{n^2}{2} + \frac{n}{2} \left( \frac{n}{2} - 1 + \sum_{j=0}^{n/2-2} (Hit\left(0, \frac{n}{2}\right) - Hit(0, j)) \right)
\]

\[
= \frac{n^2}{2} + \frac{n}{2} \left( \frac{n}{2} - 1 \right) + \frac{n}{2} \left( \frac{n}{2} - 1 \right) \frac{n^2}{2} - \frac{1}{24}(n-4)(n-3)(n-2)
\]

\[
= \Theta(nd(n^2))
\]

We can also add \(d\) back edges from each of \(k\) vertices, in the same manner, to yield \(L_3\).

**Lemma 4.10:** For natural \(d \geq 1\), and for \(k < n/2\), \(L_3\) has worst case hitting time \(Hit(0, n) = O\left((\frac{(d+1)n}{k+1})^k\right)\).

**Proof:** The proof is by induction on \(k\), and it is a straightforward combination of the analysis in the proofs of Lemma 4.8 and Lemma 4.9.

We remark that the upper bound in this theorem is not tight for non-constant \(k\), however, if we allow \(k = \Omega(n)\) in Lemma 4.8 and Lemma 4.10, the lower bounds for the worst case will be at
least $\Omega(3^{O(k)})$ and $\Omega((d+2)^{O(k)})$, respectively. In the following we will assume w.l.o.g. that $n$ is divisible by $2k$.

4.3.1 Let $G$ be a balanced Strong Chain with out-degree bounded by $D$, constructed as follows: Let vertices $v_0, \ldots, v_{n/2-1}$ be a $D$-regular Strong Chain (called the frame) and $v_{n/2}, \ldots, v_n$ be a Line digraph (called the handle). We form $G$ by uniting the frame and the handle by adding the two directed edges between $v_{n/2-1}$ to $v_{n/2}$. We call $G$ the Mirror Frame digraph because it is reminiscent of a mirror with a handle.

**Property 4.11:** Let $G$ be a Mirror Frame digraph with out-degree bounded by $D$.

1. $G$ has $Dn/2 + 2(n/2 + 1)$ directed edges.
2. $G$ has diameter $= \Omega(n)$.
3. $\text{Hit}(G) = \Theta(Dn^2)$ [Chung].

The Mirror Frame digraph is similar to the undirected lollipop graph [Feige], and it achieves the asymptotic upper bound on the hitting time for balanced digraphs. The handle part of the
digraph guarantees that the diameter is $\Omega(n)$. The best case is obtained when the frame is a $d$-expander, for constant $d$. The worst case is when the frame is a complete digraph, so that $G$ is just the digraph version of the lollipop. If $D$ is bounded by some constant, then we can get a result similar to Lemmas 4.7, 4.8, 4.9 and 4.10, since the Line digraph is basically a Mirror Frame digraph (where $D = 2$) with a “broken” frame.

**Theorem 4.12:** Let $G$ be a Mirror Frame digraph with out-degree bounded by constant $D$, $\text{Hit}(G) = \Theta(Dn^2)$.

Let $v_f$ be the frame vertex of greatest distance from $v_n$ (see Figure 4.9), so that the shortest path between $v_n$ and $v_f$, achieves the diameter of $G$. The following new digraphs are obtained by adding asymmetric back edges from the $k$ handle vertices $v_n/2, \ldots, v_{(k-1)n/2k}$ to vertices in the set $(v_f-d/2, \ldots, v_f+d/2)$.

1. Let $G_1$ be obtained by adding a single back edge to $G$. $\text{Hit}(G_1) = O(n \text{Hit}(G))$.

2. Let $G_2$ be obtained by adding a single back edge from each of $k$ distinct vertices of the handle of $G$. $\text{Hit}(G_2) = \text{Hit}(G) = O\left(\left(\frac{2n}{2k}\right)^k \text{Hit}(G) + \text{Hit}(G)\right)$.

3. Let $G_3$ be obtained by adding $d$ back edges from a single vertex of $G$, where $d < D-1$. Hit$(G_3) = \Theta(n(d \text{Hit}(G)))$.

4. Let $G_4$ be obtained from $G$ by adding $d$ back edges to each of $k$ handle vertices, where $d = D-2$. Hit$(G_4) = O\left(\left(\frac{(d+1)n}{2k}\right)^k \text{Hit}(G)\right)$.

**Proof:** We can apply the analysis of the Line digraph directly to the Mirror Frame digraph. To see this, we first transform the first half of the Line digraph $G_0$ into a cycle by adding both directed edges between $v_{n/2-1}$ and $v_0$, to form the frame. The handle is then the Line digraph from
vertex $v_{n/2}$ to $v_n$. The hitting times for the Line digraph (the handle) and the cycle (the frame) are the same. The hitting time for any frame formed by adding up to (the constant) $D-2$ edges to each vertex of this 2-regular cycle is still $O(n^2)$ (the same as the Line digraph), by Property 4.11. Back edges from the handle to the frame will be the longest back edges (edge distance $\Omega(n)$) and thus produce the greatest increase in hitting time. Adding these edges increases the hitting time by an amount proportional to the increase when adding the long back edges to the Line digraph (to $v_0$).

We conclude that we may apply the results of Lemmas 4.7 through 4.10 directly, by replacing $Hit(G_0)$ with $Hit(G)$. 

![Figure 4.10: Mirror Frame digraph](image)

**Theorem 4.13:** Let $G$ be a balanced strongly connected digraph with out-degree bounded by constant $D$. Let $G$ have diameter $= \Omega(n)$. Let $G$ have $\Omega(Dn)$ edges and $Hit(G) = \Theta(Dn^2)$.

1. Let $G_1$ be obtained by adding a single back edge from a vertex $v$, where $\text{degree}(v) < D$. $Hit(G_1) = O(n \; Hit(G))$.

2. Let $G_2$ be obtained by adding a single back edge from each of $k$ distinct vertices of $G$. $Hit(G_2) = O\left(\left(\frac{2n}{k+1}\right)^k \; Hit(G) + Hit(G)\right)$. 

45
3. Let $G_3$ be obtained by adding $d$ back edges from a single vertex $v$ of $G$, where $d \leq D - \text{degree}(v)$. $\text{Hit}(G_3) = \Theta(n(d \text{Hit}(G)))$.

4. Let $G_4$ be obtained from $G$ by adding $d_i$ back edges from each of $k$ distinct vertices $v_i$ of $G$, for $i = 1$ to $k$, where $d_i < D - \text{degree}(v_i)$.

$$\text{Hit}(G_4) = O\left(\left(\frac{(D-1)n}{k + 1}\right)^k \text{Hit}(G)\right).$$

**Proof:** The proofs of statements 1-4 are by induction, and they follow the proofs of Lemmas 4.7 to 4.9, with $\text{Hit}(G_0)$ replaced by $\text{Hit}(G)$. We begin by labeling the vertices of $G$ as $v_0$ through $v_n$, in the following manner: We select $v_0$ and $v_n$ so that the shortest path from $v_0$ to $v_n$ (the distance) is maximal (is equal to the diameter of $G$). We iteratively label the vertices from $v_{n-1}$ to $v_1$. In step $k$, we have labeled $k-1$ vertices as $v_n$ through $v_{n+k-1}$, and from the unlabeled vertices we select a vertex $v$ of maximal distance from $v_0$ and label it as $v_{n+k}$.

We may now proceed with the proofs by induction by using this labeling of $G$ in the same manner as the (natural) labeling of the Strong Chains. By our construction of the labeling of $G$, and by the assumption that the diameter of $G$ is $\Omega(n)$, distances between corresponding pairs of vertices of $G_0$ and $G$ will be proportional, that is, the distance between $v_i$ and $v_j$ in $G$ will be at least $c_1|i-j|$ and at most $c_2|i-j|$, for some constants $c_1$ and $c_2$. In the inductive step of each of these proofs, we have use the fact that adding a back edge to the Line digraph increases the hitting time at most (or at least) by an amount that is a multiplicative function of the distance between the incident vertices. Substituting the corresponding distances for $G$ will yield the stated results.

We remark that if out-degree of $G$ is bounded by some constant, the hitting time is affected only by the number of vertices that are made unbalanced by the addition of the new edges. (i.e., the size of $k$ in Theorem 4.13). We will use a similar measure of how “far” a digraph is from being balanced in the next section.
4.4 Categorizing Strong Chains

4.4.1 Harp Directed Graphs

A Harp is a directed graph whose vertices can be divided into two parts: a balanced part and an asymmetric part. The balanced part is called the head or body and the asymmetric part is called the tail or string.

The Harp, Harp(k, ℓ), is a Strong Chain with a balanced part (body) containing k vertices and an asymmetric part (tail) containing ℓ vertices, where ℓ + k = n+1, the number of vertices in the Strong Chain. The vertices of body are labeled b₀ to bₖ₋₁ and the vertices of the strings are labeled s₀ to sₗ₋₁. Each vertex of the strings is incident on two directed edges: a to next edge and a back to root edge. There is no directed edge from any vertex of body to the vertices of strings.

If the body consists of the first k vertices v₀...vₖ₋₁ then the Harp is called simple.

Lemma 4.14: Hitting Time of the simple Harp(k, ℓ) = Θ(kᶜ2ᶜ), for some constant c ≥ 1.

Proof: From the results of [Feige] and [Chung] we know that the hitting time of the body part is Θ(poly k) (we can force it to be at least linear, for example the cycle). Replacing the edge from v₀ to v₁ in the Harp Strings digraph of Lemma 4.2 by the body of Harp(k, ℓ), yields a simple recurrence, with a solution that establishes Hit(Harp(k, ℓ)) = Θ(kᶜ2ᶜ). □

In this chapter we will consider only simple Harps. In the following definitions the parameters k, ℓ, and τ are assumed to be functions of n = k + ℓ, the number of vertices in the Harp.
4.4.2 Simple Harp Families - Harp($k, \ell, \tau$)

These are the families of simple Harps with hitting times that are bounded from above by $\tau$.

Using Lemma 4.14 we have:

**Theorem 4.15:** The following bounds on $\ell$ hold:

1. For Harp($k, \ell, \text{poly}(n)$), $\ell$ is $O(\log n)$.

2. For Harp($k, \ell, n^{\log n}$), $\ell$ is $O(\log^2 n)$.

3. For Harp($k, \ell, 2^n$), $\ell = \omega(\log^2 n)$, that is, for all $c > 0$, $\lim_{n \to \infty} \frac{\ell}{c \log^2 n} = \infty$. 

Figure 4.11: An example of a simple Harp
4.5 Parameterized Complexity of Strong Chains

Let $G$ and $G'$ be Strong Chains. We say that $G'$ dominates $G$ if $\text{Hit}(G') = \Omega(\text{Hit}(G))$. A family of Strong Chains is a collection $F = \bigcup S_n$, for all $n > 0$, where each $S_n$ is a set of $n$-vertex Strong Chains. We say that a family $F' = \bigcup S'_n$ dominates family $F$ if for all $n$, there exists an $m = \text{poly}(n)$, so that for all $G \in S_n$, every $G' \in S'_m$ dominates $G$.

We define Maximal Balanced Subgraphs as followed: Let $G = (V, E)$ be a digraph. Let $G' = (V, E') \subseteq G$ be balanced. If for all balanced $G'' = (V, E'') \subseteq G$, $|E'| \geq |E''|$ then $G'$ is a maximal balanced subgraph of $G$. Let $G$ be a Strong Chain, and $G' = (V, E') \subseteq G$ be balanced with $F_0 \subseteq E'$. We say that $G'$ is a maximal balanced subgraph with base if $F_0 \subseteq E'$ and for all balanced $G'' = (V, E'') \subseteq G$ such that $F_0 \subseteq E''$, we have that $|E'| \geq |E''|$.

Let $G'$ be a maximal balanced subgraph of $G$. If vertex $v$ has more out edges in $G$ than in $G'$, then we call $v$ an asymmetric vertex relative to $G'$.

The reason for this definition is that the edges that are not in $G'$ and the number of vertices that they are incident upon as out edges, determines the worst case complexity of $G$. Note that a balanced vertex may become asymmetric relative to a choice of maximal balanced subgraph. The number of asymmetric vertices, however, is independent of the choice of maximal balanced subgraph.

**Theorem 4.16**: For any Strong Chain $C$, $|V_c| = n + 1$, $|E_c| = m$. Let $d_i$ be the out-degree of $v(i)$ in $C$. There is a simple Harp $H(k, \ell)$ that dominates $C$, where $k = n + 1$, and $\ell = \sum_{i=0}^{n-1} \lceil \log d_i \rceil < m$.

**Proof**: We assume, without loss of generality, that $C$ contains a back edge from $v_n$ to $v_0$, as this edge will clearly have no effect on the hitting time. This edge will now insure that the base is part of a balanced subgraph of $C$. 
Let $C_b$ be a maximal balanced subgraph of $C$ with base. Let $H$ be the simple Harp obtained by adding $\ell = \sum_{i=0}^{n-1} \lceil \log d_i \rceil$ string vertices to $C_b$. From Lemma 4.6 we have $\text{Hit}(C) = O(n \prod_i d_i) = O(n^2 \sum_i \lceil \log d_i \rceil) = O(n2^\ell)$. By Lemma 4.14, $\text{Hit}(H) = \Theta(2^\ell \text{Hit}(C_b))$. Thus we conclude that $\text{Hit}(H) = \Omega(2^\ell \text{Hit}(C_b)) = \Omega(2^\ell n) = \Omega(\text{Hit}(C))$. □

The construction of Theorem 4.16 is somewhat pessimistic, since we only needed to show the existence of a dominating Harp. The number of string vertices required to dominate a Strong Chain is a function of the number of asymmetric vertices. A technical issue is that a maximal balanced subgraph of $C$ will not, in general, be strongly connected. However, this is easily remedied by adding just enough edges.

**Theorem 4.17:** Let $C = (V_C, E_C)$ be a Strong Chain with out-degree bounded by some constant $D$. There is a simple Harp $H(k, \ell)$ that dominates $C$, where $k = |V_C|$, and $\ell = |V_A| \log \left[ \frac{(D - 1)^{|V_C| - 1}}{|V_A| + 1} \right]$, where $V_A$ is a set of asymmetric vertices of $C$.

**Proof:** Let $Bal$ be a maximal balanced subgraph of $C$. If $Bal$ is a strongly connected digraph, then let $G$ equal $Bal$. Otherwise, let $G$ be a balanced connected digraph obtained as follows:

Let $E_F$ be a minimal subset of $F_0$ in $C$, such that $Bal \cup E_F$ is a connected digraph.

Let $E_b$ be a minimal set of back edges such that $G = Bal \cup E_F \cup E_b$ is a balanced strongly connected digraph. Note that $|E_F| \leq |V_C| - 1$ and $|E_b| \leq |E_F|$.

Let $C' = C \cup E_b$. Clearly $\text{Hit}(C) \leq \text{Hit}(C')$. Let $V_A$ be the set of asymmetric vertices of $C'$ relative to $G$. Let $H$ be the simple Harp obtained by adding $\ell$ string vertices to $G$, where $\ell = |V_A| \log \left[ \frac{(D - 1)^{|V_C| - 1}}{|V_A| + 1} \right]$.

Let $j = |V_A|$ and $n = |V_C| - 1$, then $\ell = j \log \left[ \frac{D - 1}{j + 1} \frac{n}{j + 1} \right]$. From Theorem 4.13 we have
\[ \text{Hit}(C') = \mathcal{O}\left(\left(\frac{(D-1)n}{j+1}\right)^j \text{Hit}(G)\right) \]
\[ = \mathcal{O}(2^{j \log\left(\frac{(D-1)n}{j+1}\right)} \text{Hit}(G)) \]
\[ = \mathcal{O}(2^\ell \text{Hit}(G)). \]

By Lemma 4.14, \( \text{Hit}(H) = \Theta(2^\ell \text{Hit}(G)) \). Thus we conclude that
\[ \text{Hit}(H) = \Omega(2^\ell \text{Hit}(G)) = \Omega(\text{Hit}(C')) = \Omega(\text{Hit}(C)). \]

The number of asymmetric vertices thus gives upper and lower bounds for general digraphs. From the proof of Theorem 4.17 we conclude

**Theorem 4.18:** The family of \( n \)-vertex strongly connected digraphs with bounded maximum in-degree and out-degree \( d+1 \) and \( \ell \) asymmetric vertices has worst case cover time \( \mathcal{O}(n (n - \ell) d^\ell) \).

**Theorem 4.19:** The family of \( n \)-vertex strongly connected digraphs with maximum in-degree and out-degree \( d+1 \) and \( \ell \) asymmetric vertices has worst case cover time that is \( \Omega((n - \ell) d^\ell) \) and \( \mathcal{O}(p(n)(n - \ell) d^\ell) \) for some fixed polynomial \( p \). This gives:

1. If \( \ell = \mathcal{O}(\log n) \) the worst case cover time is \( \Theta(n^{O(\log d)}) \), which is \( \Theta(\text{poly}(n)) \) for constant \( d \).
2. If \( \ell = \omega(\log n) \) the worst case cover time is super-polynomial.
3. If \( \ell = \Theta(\log^2 n) \) the worst case cover time is \( \Theta(n^{O(\log d \log n)}) \), which is \( \Theta(n^{O(\log n)}) \) for constant \( d \).
4. If \( \ell = \omega(\log^2 n) \) the worst case cover time is \( \Theta(d^{\text{poly}(n)}) \).

**Proof:** We will modify the simple (out-degree 1) ordered cycle to obtain a bounded degree digraph with the same behavior as the Harp digraph. We add \( d \) back edges to each of the last \( \ell \) vertices of the cycle, with the destinations distributed evenly among first \( \ell \) vertices of the cycle.
The resulting digraph will have in-degree and out-degree bounded by $d+1$. Since the hitting time is less than or equal to the cover time the lower bound now follows directly from the previously mentioned Bernoulli bound [CS].

Now recall the proof of the previous theorem. The Harp $H$ that is constructed to dominate a digraph $G$ satisfies \( \text{Hit}(H) = O(2^\ell \text{Hit}(G')) \), where $G'$ is a maximal balanced subgraph of $G$. The hitting time of any balanced digraph is bounded by a small degree polynomial (e.g., we can find one that is $O(n^d)$). Since for any digraph $G$ $\text{Cover}(G) \leq n \text{Hit}(G)$ this yields the upper bound.

Since the hitting time and cover time for simple Harps are equal we can divide digraphs into classes based on their cover times using the Harps:

Consider all digraphs that are strongly connected, have bounded degree, and have a pseudo-topological order. We divide them into the following three families:

1. **WS[1]** (Weakly Symmetric[1]) – those dominated by $\text{Harp}(k, \ell, \text{poly}(n))$.
2. **WS[2]** (Weakly Symmetric[2]) – those dominated by $\text{Harp}(k, \ell, n\log n)$.
3. **WS[3]** (Weakly Symmetric[3]) – those dominated by $\text{Harp}(k, \ell, 2^{\text{poly}(n)})$.

WS[1] is contained in RL ($\cong L$), and WS[3] is contained in NL ($\subseteq L^2$). We conjecture that WS[2] is an intermediate class, between RL and PL. We could, of course, refine this hierarchy, but we are primarily interested in polynomial versus non-polynomial hitting times.
Chapter 5

Markov Chain Monte Carlo (MCMC) methods

In this chapter we discuss the relationship among random walks arising from several different applications areas. Principally, we relate the study of random walks on graphs and digraphs to the random walks arising from certain problems that are solved using Monte Carlo methods. As we have mentioned above, the primary theoretical application for the study of random walks on graphs has been to computational complexity theory. There is a rich relationship between random walks on graphs and space bounded complexity classes. The major open and solved problems in the area of log space bounded complexity classes can all be restated as problems in graph search and, in particular, randomized graph search ([NW], [ATWZ], [Reingold], [Feige], [CCZ], [CRV]). Random walks can be easily limited to log space, i.e. memory, usage during a graph search, since they only have to remember the current vertex, which takes log n bits. In contrast, a naïve deterministic search would have to remember some history of the visited vertices, using at least linear space in the size of the graph.

Practical applications of random walks on graphs have been to percolation problems and to electrical networks. Algorithms that use pseudo-random number generators are, of course, correctly regarded by complexity theorists as deterministic for a given seed value even if the seed is selected by some “random” process. In random walk on digraphs, it is assumed that we use truly random bits. While uniformly random bit sequences can now be precisely and uniquely
defined, thanks to the work in the foundations of probability theory by Kolmogorov, Martin-Löf, Chaitin, Solvay, and others [Calude], for our purposes it is not necessary. In complexity theory, one can replace probability by simple counting.

Complexity studies abstract machines, and typically uses the Turing machine model. Some of these probabilistic machines are acceptors: they either accept or reject an input (they recognize sets of inputs or languages). The input is provided on a read-only input tape and the space usage is only counted for the work tapes. A probabilistic machine will uses each random bit to choose one of two possible successor states in the computation. We assume that our machines use a polynomial (in the input size) number of random bits. Let us say that on an input the machine uses \( m \) random bits. Each of the \( 2^m \) possible bit strings represents a computation path. A technical issue here is that some paths may terminate before using all \( m \) random bits, but we can finesse this issue so that we may assume that each of the \( 2^m \) patterns of bits represents a unique computation path. The probability of acceptance is just computed as the discrete ratio of accepting paths over all paths = \#acc/2^m. \( L \) is the class of languages (acceptance problems) accepted by deterministic log space machines (no random bits), \( RL \) is the class accepted by machines that accept valid inputs with probability > \( \frac{1}{2} \), and accept invalid inputs with 0 probability (1 sided error). BPL is the class accepted by machines with bounded two sided error: valid inputs are accepted and invalid inputs rejected with probability > \( \frac{2}{3} \). This class corresponds to Monte Carlo algorithms, and the error can be made exponentially small by repeating the algorithm some polynomial number of times. It has been shown [CS78] that if one could actually use \( O(poly(n)) \) Kolmogorov-Chaitin-Solvay (KCS) [Calude] random bits then any \( RL \) or \( BPL \) machine can be made error free. However determining if a sequence is KCS (truly) random is not computable. Nevertheless there is strong evidence that \( BPL = L \). For example if \( SAT \)
requires exponential sized circuits (as is believed) then $L = BPL$. NL, nondeterministic log space, is the class accepted by machines that only have to have 1 accepting path, or nonzero acceptance probability.

Each of the well studied log space bounded complexity classes can be transformed (in deterministic log space) into an equivalent class of graph search problems. $L$ corresponds to undirected graph search, $RL$ and $BPL$ correspond to randomized search of certain well behaved digraphs and NL corresponds to search of general digraphs. One may ask why one doesn’t just use pseudo-random or quasi-random bits for these probabilistic machines. Replacing the random bits by a PRG that uses a $O(\log n)$ sized random seed would show that $L = BPL$. Perhaps surprisingly, no one has been able to do this, even though many top theorists (including Turing Award winners) have tried. The best result so far is the so-called Nissan-Widgerson generator [NW] which can “fool” random $O(\log n)$ space machines, by a truly random seed of $O(\log^{3/2} n)$ bits, thus showing BPL is in deterministic $O(\log^{3/2} n)$ space. It has also been shown [ATWZ] that $RL$ (one-sided error) is in deterministic $O(\log^{4/3} n)$ space.

Much was known about random walks on undirected graphs and these results predated the discovery by Omer Reingold, described in Chapter 3. Reingold [Reingold] showed that any undirected graph can be searched deterministically using only $O(\log n)$ work space. To appreciate how surprising this result is consider that it takes log n bits just to write down a single vertex! This established that $SL = L$, where is $SL$ is the class accepted with nonzero probability by symmetric log space bounded machines; machines where each step is reversible. Perhaps another way of viewing this result is that certain discrete ergodic Markov Chains can be simulated deterministically, with no loss of efficiency. Recall that a Markov Chain is ergodic if the underlying graph is strongly connected and it is aperiodic. In some sense all of the Markov
Chains used in computer simulations are discrete since they generally use finite precision floating point numbers. While the complexity theoretic motivation for studying random walks on undirected graphs has evaporated, the results obtained are still applicable to discrete ergodic Markov Chains. Following Reingold’s proof that undirected graph search is in $L$, the research focus has shifted to random walks on directed graphs, in an attempt to clarify the relation between the classes $L$, $RL$, and $NL$ (and others).

We can apply some of the previously known results on random walks on undirected and directed graphs, and also use some of the new results of this thesis in this area. In some discrete cases, for example in simulated annealing as applied to Sudoku puzzles, random walks can potentially have exponential time behavior. This is because the number of possible states is extremely large compared to the input size. However, even for some very simple Markov chains, one can get exponential expected time for a random walk to reach a goal state. Our results imply that even when the number of states is bounded by the problem size, the number of transitions from any state is bounded by two, the probability of each transition is at least $\frac{1}{4}$, and a lazy random walk (i.e. the walk stays in the same state with probability $\frac{1}{2}$) converges rapidly to a stationary distribution, one can still get exponential behavior. For simple classes of strongly connected and bounded degree digraphs, one can have exponential hitting time (time to reach a goal state) and exponential cover time. Random walks on graphs are basic Markov processes with a fixed number of states, and, conversely, every such Markov process can be represented by a fixed size graph. In this chapter we apply the results for randomized graph search to the random walks arising in Monte Carlo methods applied to optimization (search) problems. We relate our results to the expected hitting time of a goal state.

Additionally we relate the graph theoretic results to the use of a Monte Carlo method to
construct an accurate population sample of a Markov Chain. The results of the cover time for
directed graphs can be modified to obtain bounds on the mixing time for certain Markov Chain
Monte Carlo (MCMC) methods applied to discrete problems domains.

5.1 Degree of Balance

We demonstrated in Chapter 4 that there exist some strongly connected digraphs with
bounded out-degree $d+1$ and $\ell$ asymmetric vertices with cover time $\Omega ((n - \ell)d^\ell)$. Recall that all
strongly connected digraphs with bounded out-degree $d+1$ and $\ell$ asymmetric vertices have cover
time $O ((n - \ell)d^\ell)$. The worst case behavior for each family:

1. If $\ell = O(\log n)$ cover time is $\Theta(n^{O(\log d)}); \Theta(poly(n))$ for constant $d$.
2. If $\ell = o(\log n)$ cover time is super-polynomial.
3. If $\ell = \Theta(\log^2 n)$ cover time is $\Theta(n^{O(\log d \log n)})$.
4. If $\ell = \omega(\log^2 n)$ cover time is $\Theta(d^{poly(n)})$.

These results can be applied to discrete non-reversible Markov chains. Our random walks
correspond to Markov chains in which the probability of following an edge from vertex $v$ is just
$1/d(v)$, where $d(v)$ is the out-degree of $v$. Thus each state transition has non-negligible probability
(at least $1/n$). Conversely, we can construct an $n$ vertex weighted digraph $G(M)$ from an $n$ state
Markov chain $M$, in the usual way. We can recover unit weights by the addition of multi edges in
the natural way. This allows us to apply our notion of balance to $M$. We identify the asymmetric
states of $M$ with the asymmetric vertices of $G(M)$. The proofs of Theorems 4.17 and 4.19 in
Chapter 4 and [CCZ ] give us a nice bound for certain families of nearly balanced Markov chains.

Definition 5.1: Let $poly(x, y)$ denote some bivariate polynomial in $x$ and $y$. We measure
convergence in a given norm (typically the $\ell_2$ norm) to within some error bound $\varepsilon$. The mixing
Theorem 5.2: Let $M$ be a finite $n$ state Markov chain with graph $G(M)$. Let $G(M)$ be strongly connected with bounded degree $d$. If $M$ has at most $O(\log n)$ asymmetric states then the cover time of $M$ is polynomial in $n$. Since the mixing time of $M$ is bounded by $\text{Cover}(M)$, this implies $M$ is rapidly mixing. If $G(M)$ has degree $O(n)$ and $M$ has $c$ asymmetric states for some constant $c$ then the cover and mixing times of $M$ are polynomial in $n$.

In chapter 4, we show how to modify the computation graph of an NL machine $M$, so that we can search from vertex $s$ (the starting state of $M$) for the vertex $t$ (the accepting configuration of $M$) only on a single strongly connected (and pseudo-topologically ordered) digraph component $G$. The set of all vertices reachable from $s$ forms a strongly connected digraph $G$ of bounded degree. Lemma A.1 of [CRV] provides a method of showing that a digraph is rapidly mixing. In particular if a polynomial length $\frac{1}{2}$ lazy random walk started from any vertex returns to a given vertex $s$ with high probability, then the graph is rapidly mixing. This enables one to show that $G$ is rapidly mixing. This is because $G$ has the property that all paths are at most polynomial in length before they return to $s$. We further observe that the special class of strongly connected digraphs called harps, which we use in our proofs in [CCZ] and in Chapter 4 to obtain lower bounds, also have this property, and thus are also rapidly mixing. This allows us to conclude the following.

Theorem 5.3: There exist families of rapidly mixing finite Markov chains $M$, with $n$ states, with all state transition probabilities $> 1/c$, for constant $c$, with a goal state $t$, and $\ell$ asymmetric states where the hitting time of $t$ is $\Omega(c \ell)$, and thus exponential in $\ell$.

Thus a randomized algorithm to find an optimum state can be very badly behaved if the
underlying Markov chain is badly balanced, in the sense that we have defined above.

5.2 Markov Chain Monte Carlo (MCMC) Methods

Markov chain Monte Carlo methods are a class of algorithms for sampling from probability distributions based on constructing a Markov chain with a desired target stationary distribution. The state of the chain after a number of steps is used as a sample of the target distribution. The number of steps will affect the quality of the sample, and thus the difference from the target distribution. A good chain will have rapid mixing behavior, but this isn’t always the case. The difficult problem is to determine how many steps are needed to converge to the stationary distribution within an acceptable error. This is referred to as the mixing time. In some cases this mixing time can also be regarded as the hitting time for a goal, or target state. Random walk methods are a kind of random simulation for Monte Carlo methods; a random walk is used to reach the sample state. Random walk methods are easy to implement and analyze, however, it may take a long time (e.g. a walk with loops) to reach the target state. Therefore, some algorithms use self-avoiding random walks to prevent reentry the visited states. We remark that from the complexity theoretic viewpoint, these self-avoiding walks are closely related to unambiguous computation, and the complexity classes UL and ReachUL.

Suppose that we are given an input of STCONN, $G$, where $G$ is strongly connected, and where the smallest $\pi_i = 1/r$. We can obtain a regular digraph $G'$ by blowing up $G$, where the size of $G'$ is $\Theta(r)$ [Reingold]. Randomly choosing $k$ vertices from $G'$ produces the same sample as taking $k$ short random walks on $G$. Let the optimal solution be the probability that the walk will output the correct answer after some finite but unbounded number of steps, which, of course,
cannot be greater than 1. If the duration of the walk is not long enough, the probability that we output correct answer will less than in the optimal solution, that is, an error will be created.

In order to present our results we need to define the \textit{GAP} function as used by complexity theorists. The \textit{GAP} function is usually the absolute value of the difference between the number of accepting and rejecting paths of a nondeterministic or probabilistic Turing machine. For our purposes we will define the \textit{GAP} to be the ratio of this absolute value and the total number of paths. We may assume that a nondeterministic Turing machine accepts if and only if at least one path accepts, and always gives the correct answer, while probabilistic machines may make errors. For our purposes we can view the \textit{GAP} of a probabilistic decision algorithm \(A\) as follows: The \textit{GAP} of \(A\) is \(g(n)\) if and only if on any input \(|x| = n\) when the correct answer is YES the probability that \(A\) accepts \(x\) is at least \(\frac{1}{2} + g(n)\) and when the correct answer is NO the probability that \(A\) rejects \(x\) is at least \(\frac{1}{2} + g(n)\).

Let us use a voting analogy for clarification. Let \(A\) be a voting system, and for input \(x\), \(A(x)\) accepts if the majority vote for YES and \(A(x)\) rejects if the majority vote for NO. The \textit{GAP} function is the ratio of absolute value of the difference between the number of YES and NO voters and the total number of voters. Let \(M\) be a Monte Carlo simulation for the voting system, \(M(x)\) predicts the results of \(A(x)\) by sampling of voters. The error is the probability that \(M(x) \neq A(x)\).

\textbf{5.2.1 The \(n\)-vertex digraph \(D\) of a Probabilistic Turing Machine}

Assume that we are given a computation of a Probabilistic Turing machine \(M\) on input \(x\). Let \(n\) be the size of the set of configurations of this computation \(M(x)\) and let \(t = t(|x|)\) be a polynomial time bound on the computation time of \(M\) on input \(x\). We may assume without loss of
generality that each valid nonterminal configuration has two transitions, each with probability $\frac{1}{2}$.

By standard complexity theoretic techniques (adding a clock counter to the machine), we may also assume that each computation path terminates after at most $t$ steps. We obtain a digraph $D$ by first forming a graph $G$ as follows:

- Let the vertices represent the configurations of $M(x)$.
- Let the edges represent the valid transitions of $M(x)$.
- We add 2 self edges to each vertex representing a nonterminal configuration, so that any random walk on $D$ will be $\frac{1}{2}$ lazy.
- We add 4 self edges to each vertex representing a terminal configuration (in order to maintain the regularity of the graph).
- Vertex $s$ is the start configuration.

$D$ is then the subgraph of $G$ consisting of all vertices reachable from $s$.

**5.2.2 Transforming $D$ to a 4'-1-vertex complete 4-tree $D'$**

We expand $D$ to the tree $D'$ as follows: For every vertex $v$ of $D$ we do the following. If $v$ represents a nonterminal configuration we label the 2 edges from $v$ which correspond to transitions of $M(x)$ by 0 and 1 respectively, and the two self edges of $v$ by 2 and 3 respectively. If $v$ represents a terminal configuration then we label the 4 self edges by 0,...,3. We label each vertex by the pair $(v, p)$, where $v$ is a configuration (vertex of $D$) and $p \in \{0,1,2,3\}^*$ is a string representing a path from $s$ to $v$ length at most $t$, so that $s$ is labeled $(s, \eta)$, where $\eta$ is the empty string. We include the edge $(v, p)$ to $(w, p')$ iff $|p| < t$, $D$ contains an edge from $v$ to $w$ labeled $\sigma$, and $p' = p\sigma$. 
5.2.3 κ-vertex Sample Set $S$

We shall identify the vertex $(v, p)$ of $D'$ with the vertex $v$ of $D$, in the obvious way. Let $\varphi = 4t/2$ (the total number of leaves of $D'$), and $\kappa = n^c \geq t$, where $c$ is a constant. Let a κ-vertex sample set $S$ be a set of vertices chosen uniformly from the leaves of $D'$. We say that a vertex $v$ of $D$ occurs in $S$ (or is chosen) if $(v, p) \in S$ (or is chosen).

**Lemma 5.4:** Let $n_{D'}(v)$ be the number of occurrences of vertex $v$ in the leaves of $D'$ and $n_S(v)$ be the number of occurrences of $v$ in a κ-vertex sample set $S$. We claim that the expected value of $n_S(v)$ is $(n_{D'}(v)/\varphi)\mid S\mid = (n_{D'}(v)/\varphi)\kappa$.

**Proof:** If we choose a vertex uniformly from the leaves of $D'$ then the probability that a vertex $v$ is chosen is the probability that a vertex of the form $(v, p)$ is chosen. This is just $n_{D'}(v)/\varphi$. It follows that the expected value of $n_S(v)$ is $(n_{D'}(v)/\varphi)\kappa$. □

In what follows we assume probability of following each edge from a vertex $v$ during a random walk is $1/d(v)$ and the random walk (Markov process) shall always begin at $s$.

**Lemma 5.5:** The random walk on $D$ has a unique stationary distribution $\pi$ and the walk is rapidly mixing.

**Proof:** It is well known that a ½ lazy walk on a out-regular digraph has a unique stationary distribution. Moreover this walk is rapidly mixing [CRV]]. To see this note that the tree is a complete 4-tree with depth $t$ and the mixing time for a random walk on a complete bounded degree tree is proportional to the depth of the tree. This follows immediately from the fact that such a tree is an expander of size $4^t$ and the mixing time for an $n$ node expander is $O(\log n) = O(\log 4^t) = O(t)$. We have assumed that $t$ is polynomial in the input size. □

**Lemma 5.6:** The probability that a random walk of length $t$ on $D$ ends at $v$ is $n_{D'}(v)/\varphi$, Moreover, $\pi_v = n_{D'}(v)/\varphi$.  

62
**Proof:** Note that by the out-regularity of $D$, each path of length $t$ is equally likely. By our construction each path of length $t$ in $D$ is represented by a unique leaf node in $D'$. Thus the probability of a walk ending in $v$ is just the proportion of times $v$ occurs among the leaf nodes of $D'$, and is thus $n_{D'}(v)/\varphi$. We have also assumed that the machine $M$ always reaches a terminal configuration in at most $t$ steps. Therefore, continuing the walk for any $t' > t$ steps will leave the probability of ending at vertex $v$ unchanged. □

**Theorem 5.7:** Suppose we take $\kappa t$-step random walks on $D$. Then the expected number of walks that end at a given vertex $v$ is the same as the expected value of $n_S(v)$ of a $\kappa$-vertex sample set $S$ of the leaf nodes of $D'$.

**Proof:** From the previous lemma the probability that a $t$-step random walk reaches $v$ is $n_{D'}(v)/\varphi = \pi_v$. Thus the expected number of walks that end in $v$ is $\kappa \pi_v = \kappa(n_{D'}(v)/\varphi)$. The probability that a vertex $v$ is chosen from the leaf nodes of $D'$ is $n_{D'}(v)/\varphi$. Thus the expected value of $n_S(v)$ is also $\kappa(n_{D'}(v)/\varphi)$. □

In order to prove our main result about MCMC processes we will view the sampling by random walks as a voting process. We need the following results which give an error bound on the voting process, based on the GAP and the number of samples (voters). In this way we can show how large the number of walks $\kappa$ must be to insure small error. This result was shown in [CH] as part of an analysis of the convergence rate of the Condorcet Jury theorem. They use the following result due to Cramer [DZ].

**Lemma 5.8:** If $Y_1, Y_2, \ldots, Y_n$ are independent and identically distributed Bernoulli ($p$) random variables, i.e. they are 1 with probability $p$ and 0 with probability $1-p$, and $\hat{Y}_n$ denotes their average, then for each $b > 0$,

$$P(|\hat{Y}_n - p| > b) < 2e^{-2b^2n}$$
**Lemma 5.9:** If we have $n$ independent voters who each vote correctly with probability $p = \frac{1}{2} + a$ and we accept the majority decision as the correct vote, then the probability of making an incorrect decision (error) is less than $2e^{-2a^2n}$.

**Proof:** If we count each vote $X_i$ as -1 or 1 then the sum of the $X_i$’s is positive iff the average $\bar{Y}_n$ is larger than $\frac{1}{2}$. Hence, when the probability of a correct vote from each individual is $p = \frac{1}{2} + a$, the by taking $b = a$ in the above lemma we obtain,

$$P(S_n > 0) = P(\bar{Y}_n > \frac{1}{2}) > 1 - 2e^{-2a^2n} \qed$$

**Theorem 5.10:** Let $M$ be a probabilistic machine with $GAP g(n)$. Let $V$ a Monte Carlo method that takes $\kappa$ random walks on $D$, the graph constructed from $M$ as above. If $g(n) \geq 1/t$, then if $\kappa > ct^2$

$$\text{prob}[V(x) \neq M(x)] < 2e^{-c}.$$ 

**Proof:** Substituting $\kappa$ for $n$ and $1/t$ for $a$ in lemma yields an error less than $2e^{-2(\frac{1}{2})^2ct^2} = 2e^{-c}.$ $\square$

**Theorem 5.11:** Error Bounded Probabilistic Turing machine reduces to an $n$-vertex digraph $D''$, where $\text{Cover}(D'') = O(\text{poly}(n))$.

**Theorem 5.12:** Consider the family of $n$ state MCMC process with $\ell$ asymmetric states with running times that are polynomial in the size of the input. By repeating the MCMC a polynomial number of times we can reduce the error so that:

If $\ell = k \log n$, for some constant $k$, repeating the process $c$ times the error can be made $< 1/2^c$, that is, we can make the error arbitrarily small in polynomial time.

**Proof:** For the proof note that the theorem shows that the graph corresponding to this MCMC process has polynomial cover time, for some polynomial $p_1(n)$ . This means that after running the MCMC process for $p_1(n)$ steps we reach the target vertex $v$, with probability $> \frac{1}{2}$.
Repeating this process $c$ times reduces the probability of missing the stationary distribution to less than $1/2^c$. □
Chapter 6

Many Short Walks Can Be Better Than One Long Walk

We continue our study of random walks on classes of strongly connected digraphs. We consider the question: When exploring a new neighborhood, is one long walk or many short walks a better strategy? We study the conditions under which random walks which reset to the start periodically (short walks) can replace a single (long) random walk in effectively searching a digraph. We extend our previous analysis to include weakly connected digraphs. The motivation is to develop randomized search procedures that are guaranteed with high probability to find all easily reachable vertices (poly-reachable) in a given period of time. Poly-reachable vertices are those that can be reached with high probability by a polynomial length random walk. We first transform each graph into a layer digraph. We study the probability distribution on the poly-reachable vertices (of the layer digraph) induced by the short walks. We compare the rate at which this distribution of these vertices converges to the corresponding values in the stationary distribution of a single long walk on the layer digraph. We introduce an iterative deepening random search. We use this strategy to determine the number of short walks needed to replace the single long walk, with respect to the cover time of the poly-reachable subgraph.
6.1 Layer Digraphs

Given an $n$-vertex digraph $D$ with vertices $V = \{1, \ldots, n\}$ and a distinguished start vertex $s$ (which we shall assume, w.l.o.g. is labeled 1), we construct a $k$-layered digraph $L(D)$, where $k \geq n$, with the vertices $\{1, \ldots, k\} \times V$ as follows: For each $1 \leq i < k$ there is an edge from $(i, u)$ to $(i+1, v)$ in $L(D)$ iff there is an edge from $u$ to $v$ in $D$.

![Figure 6.1 An example of a 4-layer digraph](image)

**Definition 6.1:**

1. Given a digraph $D$ and two vertices $s$ and $v$, if there is a path from $s$ to $v$, then $v$ is reachable from $s$.

2. Given a digraph $D$ and a start vertex $s$, if for all vertices $v$ in $D$, there is a path from $s$ to $v$, then $D$ is reachable from $s$.

3. Given a digraph $D$, if for each pair of vertices $(u, v)$ in $D$ there is a path from $u$ to $v$ and a path from $v$ to $u$, then $D$ is strongly connected.

From the construction for Layer digraphs one immediately obtains:

**Theorem 6.1:** If $D$ is reachable from $s = 1$, then $L(D)$ is reachable from $(1, 1)$. 

67
Theorem 6.2: For a vertex \( v \), there is a path from \( s = 1 \) to \( v \) in \( D \) iff there exists a path from \((1, 1)\) to \((i, v)\) in \( L(D) \), for some \( 1 < i \leq k \).

We examine random walks on digraphs as a means to solve the digraph reachability problem. Our goal in this section is to examine when a long random walk on a strongly connected digraph might be replaced by many short walks, without significantly changing the probability of reaching a target vertex. In the following section we will remove the strong connectivity requirement.

Definition 6.2:

1. \( \text{length(long walk)} = \text{length of a walk from } s \text{ to } t \), such that \( \text{prob(walk}_{st} = \text{yes}) \geq \frac{1}{2} \).

2. \( \text{length(short walk)} = \text{length of a walk from } s \text{ to } t \), such that \( \text{prob(walk}_{st} = \text{yes}) > 0 \).

Thus, \( \text{length(short walk)} \leq \text{length(long walk)} \). The total time from \( s \) to reach \( t \) with probability at least \( \frac{1}{2} \) by a random walk will be:

- time of a long walk \( t_{lw} = \text{length(long walk)} \).
- time of \( k \) short walks \( t_{sw} = k \cdot \text{length(short walk)} \).
- \( t_{lw}(G) = \text{Hit}(G) \).
- \( t_{lw}(s, t) = \text{Hit}(s, t) \).

Example 1:

![Graph Family 1, F_1](image_url)

Figure 6.2: Graph Family 1, \( F_1 \)

Let \( \text{length(short walk)} = n \), then
\[ t_{lw}(G) = 2^n, \]
\[ t_{sw} = 2^n \times n = 2^n + \log n. \]

Let \( b \leq c \log n \), then
\[ t_{lw}(a, b) \leq 2^c \log n = n^c, \]
\[ t_{sw} \leq 2^c \log n \times n = n^{c+1}. \]

For a time limit of \( t = \text{poly}(n) \), the set of vertices that can be reached by taking a long walk on a digraph of \( F_1 \) is the same as the set reachable by taking a few short walks.

Example 2:

![Graph Family 2, F2](image)

**Figure 6.3: Graph Family 2, F2**

Let length(short walk) = \( n \), then
\[ t_{lw}(G) = 2^2n, \]
\[ t_{sw} = 2^2n \times n = 2^{2n} + \log n. \]

Let \( a, b \) be the vertices as shown in Figure 6.3, then
\[ t_{lw}(a, b) = \frac{1}{2}(2^{n+2} + 4) = 2^n, \]
\[ t_{sw} = 4 \times n = 4n. \]

In the worst case, the hitting time for a long walk is similar to the hitting time for many short walks, as both are exponential. For the special case \( s = a \) and \( t = b \), taking short walks is actually better than taking a long walk.
6.2 Reachability

Let $A(G, s, t)$ be a randomized algorithm that given input digraph $G$ takes a random walk of length $\ell = poly(n)$ starting at $s$ and answers “yes” if the walk reaches $t$. Let $S_1$ and $S_2$ be subsets of $V$, defined as follows: The set $S_1$ and each vertex in $S_1$ are called $s$-reachable with respect to $\ell$, if $S_1 = \{ v \mid A(G, s, v) \text{ answers “yes” with probability} > 0 \}$. The set $S_2$ and each vertex in $S_2$ are called $s$-$k$-reachable with respect to $\ell$, if $S_2 = \{ v \mid A(G, s, v) \text{ answers “yes” with probability} > 1/k \}$, for some walk where $k$ is some function of $n$. For any vertex $t \in S_2$, if we run $A(G, s, v)$ $k$ times, the probability that $t$ is not reached equals $(1 - 1/k)^k \leq 1/e$, and thus the probability that $t$ will be reached is greater than $1/2$.

We now introduce an algorithm called Inductive Search which is a kind of “iterative deepening” random search.

6.2.1 $s$-$k$-reachable set ($R$)

```plaintext
current = s

for i = 1 to \ell do
    current = uniformly choose current’s out-neighbors
    if current == t
        then return “yes”
    if current == terminal
        then return “no”
end for loop

return “no”
```

Algorithm 6.1: Random Walk($G, s, t, \ell$)
let \( j = n \)

while \( j \leq n^{\log \log n} \)

for \( i = 1 \) to \( j \) do

if Random Walk\((G, s, t, j)\) == “yes”

output “yes” and halt

end for loop

\( j = j^2 \)

end while loop

output “no” and halt

Algorithm 6.2: Inductive Search\((G, s, t)\)

**Theorem 6.3:** For all sufficiently large \( n \) and \( k \leq n^{\log \log n} \):

1. For all \( v \in s\text{-reachable} \), Inductive Search will answer “yes” with probability > 0.

2. For all \( v \in s\text{-k-reachable} \) with respect to \( \ell \), Inductive Search will halt probability \( \geq \frac{1}{2} \) in time at most \( O(\max(\ell^2, k^2)) \).

3. For all \( v \not\in s\text{-reachable} \), Inductive Search answers “no”.

**Proof for 2:**

Notice that Inductive Search, upon reaching (or surpassing) a value \( x \), repeats a random walk of length \( x \), \( x \) times in the inner loop. If \( \ell \geq k \), then consider the first iteration (if reached) in which \( j \geq \ell \). The probability that Inductive Search will fail to answer “yes” during this iteration is \( \leq (1 - 1/k)^j \leq (1 - 1/k)^k \leq 1/e \), and thus \( > \frac{1}{2} \). Thus the probability that the algorithm halts and says yes before or during this iteration is \( > \frac{1}{2} \).
If \( \ell < k \), then consider the first iteration (if reached) for which \( j \geq k \). The probability that Inductive Search will fail to answer “yes” during this iteration is \( \leq (1 - 1/k)^k \leq 1/e \), and so the probability that Inductive Search will answer yes by this time is \( > \frac{1}{2} \). The running time is dominated by the final iteration executed, which has complexity \( O(j^2) \). □

Note that if \( k = \text{poly}(n) \) then the running time of Inductive Search is polynomial in \( n \). Thus, for all \( v \in \text{s-k-reachable} \), Inductive Search will halt and output “yes” with probability \( \geq \frac{1}{2} \), within polynomial time.

**Theorem 6.4:** If a vertex \( v \) is \( s\)-reachable from \( s \) and for each path, \( \text{path}(s,v) \), from \( s \) to \( v \), \( \exists u \in \text{path}(s,v) \) and \( u \) is not \( s\)-k-reachable where \( k = \text{poly}(n) \), then \( v \) is not \( s\)-k-reachable.

### 6.2.2 Poly-Reachability

**Definition 6.3:** Given an \( n \)-vertex digraph \( D \) and a vertex \( s \), we construct a \( k \)-layer digraph \( L(D) \). We let \( L = L(D) \) be the subgraph of the layer graph consisting of only the vertices reachable from \((1,1)\). Let \( M(L) \) be a discrete-time Markov Chain that represents a random walk on \( L \) which resets after \( k-1 \) steps. Let \( P(L) \) be the transition matrix for \( M(L) \) defined as for all \( 1 \leq i < k \), if \( u \) is a nonterminal state, then \( P(L)[(i,u),(i+1,v)] = P[u,v] \); otherwise, \( P(L)[(i,u),(i+1,u)] = 1 \). So that \( M(L) \) represents \( k-1 \) step random walks on \( D \). Let \( \rho(D) \) and \( \rho(L) \) be stochastic row vectors for \( M(D) \) and \( M(L) \) such that \( \rho^j(L) = \rho^{j \mod k}(D) \).

**Lemma 6.5:** The graph \( G = G(M(L)) \) of \( M(L) \) is strongly connected.

**Proof:** The walk resets after reaching the \( k^{th} \) layer of \( L \), and thus \( G \) consists of \( L \) together with a directed edge from each vertex of the form \((k,v)\) to \((1,1)\). Since by assumption, \( L \) consists of vertices reachable from \((1,1)\), this establishes the strong connectivity of \( G \). □

The construction immediately implies the following fact.
Lemma 6.6: For any pair of vertices $u$ and $v$ in $G$, the length of the shortest path from $u$ to $v$ is less than $2k$.

Definition 6.4: Given an $n$-vertex digraph $D$ and a vertex $s$, we construct a $(k+1)$-layer digraph as in definition 6.3. Let $L = L(D)$ be the subgraph of the layer graph consisting of only the vertices reachable from $(1, 1)$. Let $M(L)$ be a discrete-time Markov Chain that represents a $\frac{1}{2}$ lazy random walk on $L$. Let $P(L)$ be the transition matrix for $M(L)$ defined so that $P(L)((i, u), (i+1, v)) = \frac{1}{2}P[u, v]$ and $P(L)((i, u), (i+1, u)) = \frac{1}{2}$ for all $1 < i \leq k$, so that $M(L)$ represents $k$ step $\frac{1}{2}$ lazy random walks on $D$. Let $\rho^i(L)$ be the probability distribution after $i$ steps of a $\frac{1}{2}$ lazy random walk on $L$. Thus, $\rho^k(L)$ is the distribution for the bottom layer.

One observes

Lemma 6.7: $\forall i, \lim_{k \to \infty} \frac{\rho^i_k(L)}{\pi_i(D)} = 1$.

Proof: This follows from the properties of a lazy random walk on a digraph, see for example [Chung] Theorem 5.

Definition 6.5: Let $\pi$ and $\pi'$ be stochastic row vectors. We define several measures of closeness.

1. If $\forall i, |\pi'_i - \pi_i| \leq \alpha$, then we say that $\pi$ and $\pi'$ are $\beta$-close for $\beta = \alpha$.

2. If $\forall i, \pi'_i < \pi_i$, $\frac{\pi'_i}{\pi_i} \geq \alpha$, we say that $\pi'$ is in $\delta$-proportion to $\pi$, for $\delta = \alpha$.

Theorem 6.8: There exist some graph families, such as $F_1$, so that for any $n$-vertex digraph $D$ in these families, and for any two vertices $s, t \in D$, if one takes a $\frac{1}{2}$ lazy random walk from $s$ of $n^2$ steps, $t$ is reached with probability greater than $\frac{1}{2} \pi_t$. If one takes $3 \frac{1}{2}$ lazy random walks from $s$ of $n^2$ steps each, $t$ is reached with probability greater than $\pi_t$.

Proof: For each $n^2$-step walk, $t$ is not reached with probability less than $1 - \frac{1}{2} \pi_t$. The probability of $t$ is not reached of $3$ $n^2$-step walks is less than $(1 - \frac{1}{2} \pi_t)^3 = 1 - 3\frac{1}{2} \pi_t + 3(\frac{1}{2} \pi_t)^2 -$
$(\frac{1}{2} \pi_t)^3$. The probability of $t$ is not reached of $3 \thinspace n^2$-step walks is greater than $3/2 \pi_t - \frac{3}{4} \pi_t^2 + 1/8$

$\pi_t^3 > \pi_t <$

We can see that for graph families like $F_1$, we can replace a long walk by a few short walks and achieve the same probability of reaching the target. The question remains: can we do the same for graph families like $F_2$, which are not in $\delta$-proportion to $\pi$, for $\delta \geq 1/poly(n)$? Can we still replace a long walk by a few short walks and achieve the same probability of reaching a target vertex?

### 6.3 Directed Cheeger Constants

The Cheeger constant $h(G)$ of a directed graph $G$ bounds the “flow of probability” through $G$ (Reingold terms this “entropy waves”) and thus the convergence rate of a random walk on $G$. Thus, if $h(G)$ is large, for example $h(G) = poly(n)$, then a random walk on $G$ will converge to the stationary distribution $\pi$ in polynomial time.

**Theorem 6.9:** There exists a strongly connected $n$-vertex digraph $G$ with stationary distribution $\pi$ and with $\pi_s$ and $\pi_t$ non-negligible (greater $1/poly(n)$), with $Hit(s, t) = \Omega(exp(n))$.

**Proof:** If $h(G) = O(1/exp(n))$ then $Hit(s, t)$ might not be polynomial in $n$, even both $\pi_s$ and $\pi_t$ are non-negligible. Consider, for example, $s$ and $t$ are two end vertices in the graph of Figure 6.3 used in [Chung]. $\pi_s = \pi_t = \frac{1}{6}$ and $h(G) = (\frac{1}{6} (2^n+3))/(1/2) = 1/(3 \cdot 2^{n-1})$, $Hit(s, t) \geq (3 \cdot 2^{n-1}) > 2^n$.

We now generalize the usual notion of the Cheeger constant of a directed graph to the flow over an arbitrary cut, to yield a family of flow rate constants. The purpose of this definition is to allow us to consider the flow only over certain special cuts: those cuts that separate the two distinguished vertices $s$ and $t$. 
Definition 6.6: Given a $n$-vertex digraph $D$ and two vertices $s$ and $t$. Let $\ell = \text{poly}(n)$ and let $L$ be a $(\ell + 1)$-layer graph obtained from $D$ as defined in section 6.1. The Cheeger constant (flow rate) between $s$ and $t$ with respect to $\ell$ will be $h(s, t) = \sum_u F(u, t)$ for all edge $(u, t) \in E(L)$.

Thus, if we take $h(s, t)$ walks from $s$ of length $\ell$, then $t$ will be reached with probability greater than $\frac{1}{2}$, e.g. $\text{Hit}(s, t) \leq \ell h(s, t)$.

Let $\phi$ be the distribution of the bottom layer of $L$, and let $\pi$ be the stationary distribution of $D$. We said $D$ is $\delta$- $\ell$ -mixing if $\phi$ is in $\delta$-proportion to $\pi$ and $D$ is partial $\delta$- $\ell$ -mixing if a subset of $\phi$ is in $\delta$-proportion to $\pi$. For example, digraphs in $F_1$ are $\frac{1}{2}n^2$-mixing, and digraphs in $F_2$ is partial $\frac{1}{2}n^2$-mixing. If $h(s, t)$ is greater than $1/\text{poly}(n)$ then $\text{Hit}(s, t) = O(\text{poly}(n))$. Note that this remains true even if $h(D) = O(1/\exp(n))$. Thus, if the hitting time of a long walk is bounded by $\text{poly}(n)$, then the hitting time of the sequence of short walks is also bounded by $\text{poly}(n)$. Moreover, for some cases such as the family $F_2$ (see Figure 6.2), when starting at vertex $a$ taking short walks is more efficient. In this case for a single random walk $\text{hit}(a, b)$ is exponential, but the process of many short walks will have polynomial hitting time. If $h(s, t)$ is less than $1/\exp(n)$, then the hitting time of both a single long walk and many short walks will be exponential.

### 6.4 Bagging Vertices

Let $L$ be a $\ell$-layer digraph, where $\ell$ is some $\text{poly}(n)$. Let $\pi$ be the distribution for the bottom layer of $L$. Let $b(n)$ be some bound (function of $n$). We can obtain $L'$ from $L$ by putting some vertices in a “bag.” Let us order the vertices of $L$ according to the distribution $\pi$, in order of decreasing probability. That is, we label the vertices of $L$ by $\{1, \ldots, n\}$, so that $\pi_1 \geq \pi_2 \geq \ldots \geq \pi_\ell \geq \ldots \geq \pi_n$. 

75
1/b(n) > π_{k+1} ≥ \ldots ≥ π_n. We now “bag” the vertices \{k+1, \ldots, n\} by combining them into a single vertex (k+1) of L'. The vertices of L'={1, \ldots, k+1}.

**Lemma 6.10:** Let d be the maximal out-degree in L, then π'_{k+1} ≥ 1/db(n).

**Proof:** In the worst case, there is a single in-edge of vertex (ℓ, k+1) from vertex u in L'. Thus π'_{k+1} ≥ 1/db(n), since j has at most d out edges. □

**Theorem 6.10:** If b(n) is some polynomial in n, then the cover time for L' will be also be polynomial in n.

**Proof:** From the previous lemma π'_{i} ≥ 1/db(n), for all i. Since d is at most n and b(n) is assumed polynomial, then π'_{i} ≥ 1/poly(n). We can bound the hitting time of any vertex i by observing that π'_{i} random walks of length ℓ will visit vertex i with probability > ½. Thus Hit(L') ≤ ℓ π'_{i} ≤ ℓ db(n). Therefore, Cover(L') ≤ ℓ db(n) log (db(n)), which is polynomial in n, since ℓ and b(n) are both polynomial in n. □
Chapter 7

Conclusion and Future Work

We are currently undertaking the development of search algorithms for the digraph families introduced in this thesis. We would like to develop strategies that exploit the structure of these graphs. Given a digraph $G$, we would like to know what algorithm is most suitable for solving \textsc{STCON} on $G$. Our goal is either to perform space efficient modification of a digraph to improve its hitting time (modified random search) or to deterministically search a digraph with non-polynomial cover time using limited space.

In the first phase, structure checking, there are several special cases that can be identified in log space: balanced digraphs, Harp Strings digraphs, and Eulerian Path digraphs. For a balanced digraph, we can use a lazy random walk or Reingold’s Pseudorandom Generator (PRG) which is a derandomized algorithm using $O(\log n)$ space. For a Harp Strings digraph, we can force the walk to choose only to next edges. A Harp Graph can be searched by a hybrid of a lazy random walk (for each balanced portion), and using the deterministic “force to next strategy” (on the string portions). For an Eulerian Path digraph, we can add a single directed edge to make it balanced. For digraphs from each of the different weakly symmetric classes we will examine the results of applying modified random walks to digraphs, either by adding edges or by changing the probabilities for choosing asymmetric vertices, in order to make the graphs more “balanced”.

In the next phase of our work we will study the union of Strong Chains, the digraph families that more closely model NL computation graphs. For example, a first step is to identify the
conditions under which the non disjoint union of Strong Chains will still form a digraph with a
search complexity that matches one of our Weakly Symmetric families. This union may, in some
cases, come very close to having good expansion properties. This is because the exponentially
many distinct computation paths of a weakly symmetric NL machine are “threaded” through
only a polynomial number of distinct vertices of a sparse digraph. Finally, we observe that in the
analysis of fooling space bounded computations by the INW generator ([INW], 1994), the
computation was divided into communicating phases, but not all of the structure of computations
was actually exploited. The additional structure to exploit is that if there is a path between two
configurations of some small constant length \( k \) then the work tape contents must be the same
except for at most \( k \) cells in a region of size \( 2^k \) surrounding the tape head position. We will
attempt to exploit this fact to define a distance metric on the network and see if this additional
structure between phases allows for a tighter analysis of the number of random bits actually
required.

Applying the machinery developed to analyze random walks on graphs allows us to draw
some conclusions on the efficiency of certain Monte Carlo simulations, sampling processes, and
MCMC methods. Our previous results on random walks [CCZ] allow us to bound the running
time and error of certain Markov processes \( M \), from a bound on the cover time of the
corresponding graph \( G(M) \). In future work we will try to extend these bounds to more general
Markov processes by a more detailed examination of the relation between a Markov Chain and
its associated graph.

We have shown that taking few short random walks can be more effective than taking a
single long random walk when searching certain families of digraphs. Moreover, this strategy
cannot perform much worse than the single walk strategy on any digraph. Our future work will
examine how to more effectively manage limited resources (space and time) use these resources to reach our target (get the most profit).

Another intriguing area of investigation concerns the space bounded exploration of reach-unambiguous graphs and the class ReachUL. The relation between ReachUL, NL, and RL is unknown. While it has been shown (via oracle results) that probabilistic and unique computation are probably incomparable [EHT], in the context of polynomial time, no such results are known for log space. Moreover, the class ReachUL is defined by a different notion of unambiguity than the classes UL and UP, and thus is likely a proper subset of UL. One novel idea that we are currently examining is using many short walks to deduce digraph structure, such as path length statistics, rather than for the full exploration of the reach unambiguous graph.

![Diagram of complexity classes](image)

Figure 7.1: Complexity Implications

We are working on subclasses of UL, such as BPUL, the intersection of ReachUL and BPL and RUL, the intersection of ReachUL and RL. These classes correspond to probabilistic logspace machines with a polynomial number of computation paths from the start to any
reachable configuration. We seek to further investigate the relation between ReachUL, RL, and BPL. We are attempting to combine the idea of using short random walks to obtain path length statistics with results from graph pebbling to produce hybrid random and deterministic space bounded algorithms for searching logspace bounded computation trees. For example, can this technique be used to exhibit a randomized algorithm for the class ReachUL?
Bibliography


[CS07] LM CHAVES, DJ de SOUZA. Waiting Time for a Run of N Success in Bernoulli


[GMRZ] Parikshit Gopalan, Raghu Meka, Omer Reingold, David Zuckerman: Pseudorandom generators for combinatorial shapes. STOC 2011: 253-262


