Derandomization & Time-Space Trade-off in Efficient Computation

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Diptarka Chakraborty

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

August, 2016

CERTIFICATE

It is certified that the work contained in the thesis entitled "Derandomization & Time-Space Trade-off in Efficient Computation", by *Diptarka Chakraborty*, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

Manindra Agrawal
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Satyadev Nandakumar
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

August, 2016

Synopsis

Pseudorandomness

Incorporating randomness in any feasible computation is of fundamental importance in the field of theoretical computer science. In order to simulate any efficient randomized algorithm, we do not need pure random bits or uniform distribution. Instead, what we need is that behaves almost same as uniform distribution or in other words the distribution looks similar to uniform distribution to that algorithm. This is the basic idea of the notion called *pseudorandomness* which is defined via two provably equivalent notions known as *computational indistinguishability* and *unpredictability.*

Characterization of Distributions in terms of Pseudorandomness Once we have the definition of pseudorandomness, it is natural to ask the question how to characterize distributions on the basis of the amount of pseudorandomness present in them. Generally, to classify random sources, information theoretic notion known as *min-entropy* is used. In case of computational analogue of entropy, we have several such notions like Yao-type (or compression based) *pseudoentropy*, HILL-type pseudoentropy, next-bit pseudoentropy. However, it is not clear which of the above notions is the most appropriate or whether they are at all suitable. In this thesis, we propose an alternate characterization of distributions to quantify the amount of pseudorandomness present in them. For this purpose we adapt the theory of *dimension* defined via betting functions called *s-gales*, a generalized notion of martingales and *log-loss unpredictability.* Then we introduce the notion of "non-uniform" gale and a new probabilistic definition of success of a gale over a distribution. We show some important properties of dimension which make our definition robust. Next, we give a comparative study between our notion of dimension and different notions of pseudoentropy.

Pseudorandom Extractor After quantification of pseudorandomness of a distribution the next natural question is how to extract out pseudorandom part out of any distribution. This is analogous to the question of constructing *randomness extractor*. Unfortunately, it is not possible to construct any randomness extractor deterministically. To extract out almost all the randomness, we need extra $\Omega(\log n)$ pure random bits. It was not clear whether $\Omega(\log n)$ many extra random bits are also required to extract out only the pseudorandom part. In this thesis, we consider the following: *Given a distribution with dimension s, the problem is to output $O(sn)$ many bits that are pseudorandom.* We show $\Omega(\log n)$ lower bound on the number of extra pure random bits required to solve the above stated problem. However, constructibility of such pseudorandom extractor using only $O(\log n)$ extra random bits is still open. On the other hand, for distributions having high HILL-type pseudoentropy, any randomness extractor and in case of high Yao-type pseudoentropy, any "reconstruction type" randomness extractor will serve our purpose. Unfortunately, a distributions with high dimension may have very low pseudoentropy of either type and by our work, actually the same counterexample will work for both the cases. However, for a special case, namely for *nonpseudorandom bit-fixing sources*, we show that the construction same as that of deterministic randomness extractor for *bit-fixing random sources* will work.

Approaching towards P vs. BPP In this thesis, to the end of the first part we also show that for proving $\mathsf{P} = \mathsf{BPP}$, it is sufficient to construct an algorithm that stretches $O(\log n)$ pure random bits to n bits such that the output distribution has a non-zero dimension instead of being pseudorandom, because from such algorithm we can easily construct a hard function satisfying certain hardness requirements sufficient to build an *optimal pseudorandom generator*.

Relations among Complexity Classes L, NL and SC

In the domain of space bounded computation, one of the most important questions is whether non-determinism adds any extra power to the log-space computation or not. A much weaker question, whether NL is inside SC (simultaneous polynomial time and poly-logarithmic space class) or not is also yet to be resolved. The most obvious way to tackle this weaker question is to try for an SC algorithm for the *reachability problem* for directed graphs, which is known to be NL-complete. For this problem, the trade-off between two basic algorithms is that BFS takes linear space and linear time whereas Savitch's algorithm takes $O(\log^2 n)$ space but $\theta(n^{\log n})$ time. Wigderson posed the following question: *Can we design a polynomial-time algorithm for the directed graph reachability problem that uses only $O(n^{1-\epsilon})$ space for some small constant $\epsilon > 0$?* Till now best known algorithm is by Barnes, Buss, Ruzzo and Schieber and uses $O(n/2^{k\sqrt{\log n}})$ space, for any constant k while keeping time polynomial. However, for directed planar graphs a recent result by Imai *et al.* achieved $O(n^{1/2+\epsilon})$ space, for any $\epsilon > 0$ and polynomial time bound simultaneously. Later in a joint work with Pavan, Tewari, Vinodchandran and Yang, we extended this approach to give $\widetilde{O}(n^{2/3}g^{1/3})$ space (by $\widetilde{O}(s(n))$ we mean $O(s(n)(\log n)^{O(1)})$) and polynomial time bound for directed graphs embedded on orientable surface of genus g.

Reachability in H-minor-free Graphs In this thesis, we extend our result of high-genus graphs to H-minor-free graphs by using Robertson and Seymour's Graph Minor Decomposition Theorem and provide $\widetilde{O}(n^{2/3})$ space algorithm. Our main contribution is to provide a space efficient construction of *separator* for this particular graph class and for that purpose we use the construction of *planarizing set* used for deciding reachability in high-genus graphs. Moreover, using a similar type of decomposition theorem given by Thierauf and Wagner, we achieve the bound same as that given for directed planar graphs by Imai *et al.* for $K_{3,3}$-free and K_5-free graphs, which is a strict superset of planar graphs.

Reachability in Directed Layered Planar Graphs Just like for directed graphs, reachability problem for directed layered graphs is also NL-complete and thus giving an

SC algorithm for this problem also suffices to show NL $\subseteq$ SC. No better bound than that given for general directed graphs is known for this problem. So it is natural to study this problem under planarity restriction and under that restriction also, best known bound is same as achieved by Imai *et al.* mentioned earlier. In this thesis, we provide a polynomial time algorithm for reachability problem for directed layered planar graphs that uses only n^ϵ space, for any $\epsilon > 0$. For this purpose, we consider the reachability problem for layered grid graphs as directed layered planar case is log-space reducible to it. Finally, by using a modified DFS strategy into a courser grid structure along with a clever "marking scheme", we prove our result for this reduced problem.

Results for Red-Blue Path and other Problems in Planar Graphs It seems that problems on planar graphs are easier than that on general directed graphs. So it is natural to search for some problem that restricted to planar graphs is also NL-complete. One such candidate is *Red-Blue Path* problem for planar DAGs. In this thesis, we provide an $O(n^{1/2+\epsilon})$ space, for any $\epsilon > 0$ and polynomial time algorithm for this problem using the space efficient construction of *planar separator* given by Imai *et al.* while giving algorithm to solve directed planar reachability problem. This is the first simultaneous $O(n^{1/2+\epsilon})$ space, for any $\epsilon > 0$ and polynomial time bound known for any NL-complete problem. However, as the corresponding complexity class is not closed under log-space reductions, we do not get any containment result for the class NL. We also extend our result to several other problems like shortest path in planar graphs, even path problem in planar DAGs, perfect matching problem for planar bipartite graphs and also for even perfect matching problem in planar bipartite graphs. For none of the above problems any simultaneous sublinear space and polynomial time bound was known before.

To my parents...

Acknowledgements

I am extremely thankful to my advisors Manindra Agrawal and Satyadev Nandakumar for their continuous guidance and immense support to carry research throughout my PhD. They taught me how to climb the mountain named theoretical computer science and explore the unexplored region without the fear of slipping down and made me realize that sudden slipping down may actually lead to a wonderful valley of several beautiful yet possibly unproved theorems.

I have been very fortunate to have Manindra Agrawal as my supervisor. He gave me enough freedom to play with my own ideas. However, probably the biggest lesson I have learnt from him is not to be afraid of any problem irrespective of how scary it apparently looks like. He never asked me to solve any problem, instead told me to only understand the problem clearly, and finally that understanding has helped me to devise solution in many scenarios. I am very much fond of his way of thinking about any seemingly new problem. I wish one day I could think artistically like him. I want to thank not only him but his family as well, for inviting me on several occasions.

During my PhD I have worked with both Satyadev and Raghunath and I do not know how much thanks I should give them because no matter how many words I spend, I am sure it will fall much shorter than what they actually deserve. They both have always been very generous with their time to listen my several seemingly nonsense ideas and being patient with me. Whenever I felt low I just walked into one of their rooms and start discussing on some arbitrary topics and most of the time the topics were completely orthogonal to our research domain.

I express my gratitude to all the faculty members of CSE department of IIT Kanpur for their excellent teaching during the time of course work and also for providing me a nice friendly environment inside the department. I particularly thank Sumit Ganguly, Piyush Kurur and Nitin Saxena for all the discussions I had with them. I sincerely thank Surender Baswana for his truly amazing classes and only due to him I have learnt to listen and enjoy the music of algorithms. I also thank Somenath Biswas for inspiring me in doing research in complexity theory.

I had the opportunity to make two academic visits during the term. I would like to thank Michal for hosting me in the Charles University in Prague and Andrej for hosting me in The Chinese University of Hong Kong and also for all the discussions I had with both of them. I want to thank Elazar and Siyao for all the interesting discussions on theory and other things and I am sure without them my visit in Prague and Hong Kong would not have been as enjoyable as it actually was. I also thank my other collaborators, particularly Pavan and Vinodchandran.

Life in IIT Kanpur have been very enjoyable and full of fun due to the company I had there. Many thanks to Garima, Keerti, Jaydeep, Atanu, Biman, Rohit, Arpita, Seetha Ram and all other friends. Special thanks to Sumanta, Debarati and Amit for being incredible friends and for always being there irrespective of how I behave with them. I would not have learnt as much as I did without them.

Most of all, thanks to my parents and my maternal uncle for their love, constant support and encouragement. Without their support and encouragement this work would not have been possible.

I thank the institute as a whole for providing us such a beautiful environment. Last but not the least, I would like to thank my overall PhD time for teaching me how to be happy or at least be pseudo-happy even on failures and giving me the strength to overcome all the tough situations I faced in research as well as in personal life.

Contents

List of Publications

[ACDN15] **Dimension, Pseudorandomness and Extraction of Pseudorandomness**
with Manindra Agrawal, Debarati Das and Satyadev Nandakumar
In Proceedings of the 35th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), pages 221-235, 2015

[CT15a] **An $O(n^\epsilon)$ Space and Polynomial Time Algorithm for Reachability in Directed Layered Planar Graphs**
with Raghunath Tewari
In Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC), pages 614-624, 2015

[CT15c] **Simultaneous Time-Space Upper Bounds for Red-Blue Path Problem in Planar DAGs**
with Raghunath Tewari
In Proceedings of the 9th International Workshop on Algorithms and Computation (WALCOM), pages 258-269, 2015

[CT15b] **Simultaneous Time-Space Upper Bounds for Certain Problems in Planar Graphs**
with Raghunath Tewari
CoRR, abs/1502.02135, 2015
(This is an extended version of the paper entitled "Simultaneous Time-Space Upper Bounds for Red-Blue Path Problem in Planar DAGs" that appeared in WALCOM 2015)

[CPT$^+$14] **New Time-Space Upperbounds for Directed Reachability in High-genus and H-minor-free Graphs**
with Aduri Pavan, Raghunath Tewari, N. V. Vinodchandran and Lin Forrest Yang
In Proceedings of the 34th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), pages 585-595, 2014

Contents of Chapter 3, 4 and 5 are based on the results of [ACDN15]. Chapter 9 is based on a part of [CPT$^+$14] and Chapter 10 is based on [CT15a]. The work in Chapter 11 is based on [CT15b], a preliminary version of which is [CT15c].

List of Figures

Part I

Pseudorandomness & Derandomization

Chapter 1

Introduction

Incorporating randomness in any feasible computation is one of the basic primitives in theoretical computer science. Fortunately, any efficient (say, polynomial time) randomized algorithm does not actually require pure random bits. What it needs is a source that *looks* random to it and this is where the notion of *pseudorandomness* [BM84, Yao82] comes into picture. Since its introduction, pseudorandomness has been fundamental to the domain of cryptography, complexity theory and computational learning theory. Pseudorandomness is mainly a computational approach to study the nature of randomness, and *computational indistinguishability* [GM84] plays a pivotal role in this. Informally, a distribution is said to be pseudorandom if no efficient algorithm can distinguish it from the uniform distribution. Another way of looking at the notion of pseudorandomness is via the notion of *unpredictability* of distributions, due to Yao [Yao82]. Computational indistinguishability and unpredictability are provably two equivalent ways to define pseudorandom distributions. A distribution is *unpredictable* if there is no efficient algorithm that, given a prefix of a string coming from that distribution, can guess the next bit with a significant success probability. This line of research naturally posed the question of constructing algorithms that can generate pseudorandom distributions, known as *pseudorandom generators.* Till now we know such constructions by assuming the existence of *one-way functions.* It is well known that constructibility of an *optimal pseudorandom generator* implies complete derandomization (i.e., $\mathsf{P} = \mathsf{BPP}$) and *exponential hardness assumption* on one-way function enables us to do that. However, Nisan and Wigderson [NW94] showed that the existence of an exponential *hard function*, which is a much weaker assumption, is also sufficient for this purpose. The assumption was further weakened in [IW96].

In order to characterize the class of random sources, information theoretic notion of *min-entropy* is normally used. A computational analogue of entropy was introduced by Yao [Yao82] and was based on compression. Håstad, Impagliazzo, Levin and Luby [HILL99] extended the definition of min-entropy in computational settings while giving the construction of a pseudorandom generator from any one-way function. This definition of computational version of min-entropy is known as HILL-type *pseudoentropy* and it basically extends the definition of pseudorandomness syntactically. Relations among above two types of pseudoentropy was further studied in [BSW03] and so far it is not known whether these two types of computational analogues of entropy are equivalent or not with respect to polynomial-size circuits. Another type of definition also appears in the literature, namely metric-type pseudoentropy which is nothing but reversal of quantifiers in the definitions of HILL-type pseudoentropy. It has been shown in [BSW03] that this latter variant of pseudoentropy is actually same as HILL-type pseudoentropy with respect to the class of all polynomial-size circuits. However, for some less powerful model like bounded-width read-once oblivious branching programs they are not same. On the other hand, if we consider very powerful model of computation like PH-circuits, then all three notions are equivalent up to some small factor [BSW03]. A more relaxed notion of pseudoentropy, known as *next-bit Shannon pseudoentropy*, was later introduced by Haitner, Reingold and Vadhan [HRV10] in the context of an efficient construction of a pseudorandom generator from any one-way function. In a follow up work [VZ12], the same notion was alternatively characterized by *KL-hardness*. So far it is not clear which of the above notions is the most appropriate or whether they are at all suitable to characterize distributions in terms of the degree of pseudorandomness in it.

In this thesis, we first propose an alternative measure to quantify the amount of pseudorandomness present in a distribution. This measure is motivated by the ideas of *dimension* [Lut03b] and *logarithmic loss unpredictability* [Hit03]. Lutz used the betting functions known as *gales* to characterize the *Hausdroff dimension* of sets of infinite sequences over a finite alphabet. The definition given by Lutz cannot be carried over directly, because here we consider the distributions over finite length strings instead of sets containing infinite length strings. To overcome this difficulty, we allow "non-uniform" gales and introduce a new probabilistic notion of *success* of a gale over a distribution. We use this to define two notions of *dimension* of a distribution - strong one and a weak one. Both the notions were already there for the case of infinite strings [Lut11]. A similar approach was previously attempted to tackle the same question in [G06, Rag11, Das14]. A few results discussed in this thesis are modified version of the results stated in [Das14]. More specifically the criterion of "win" of any s-gale and as a consequence almost all the results of [Das14] had some potential drawbacks and are resolved in this thesis. In [Hit03], Hitchcock showed that the definition of dimension given by Lutz is equivalent to logarithmic loss unpredictability. In this thesis, we show that this result can be adapted to establish a quantitative equivalence between the notion of logarithmic loss unpredictability of a distribution and our proposed notion of dimension. Roughly this captures the essence of equivalence between pseudorandomness defined via indistinguishability and via unpredictability. We show some important properties of the notion of dimension of a distribution, which eventually makes this characterization much more powerful and flexible. We also do a comparative study between our notion of dimension and two provably not equivalent notions of pseudoentropy, namely HILL-type pseudo min-entropy and next-bit pseudo Shannon entropy. We show that the class of distributions with high dimension is a strict superset of the class of distributions having high HILL-type pseudo min-entropy. Whereas, there is a much closer relationship between dimension and next-bit pseudo Shannon entropy.

Once we have a quantification of pseudorandomness of a distribution, the next natural question is how to extract the pseudorandom part from a given distribution. The question is similar to the question of constructing *randomness extractors* which is an *efficient* algorithm that converts a realistic source to an *almost* ideal source of randomness. The term *randomness extractor* was first defined by Nisan and Zuckerman [NZ93]. Unfortunately there is no such deterministic algorithm and to extract out almost all the randomness, extra $\Omega(\log n)$ pure random bits are always required [NZ96, RTS00]. There is a long line of research on construction of extractors towards achieving this bound. For a comprehensive treatment on this topic, we refer the reader to excellent surveys by Nisan and

Ta-Shma [NT99] and Shaltiel [Sha02]. Finally, the desired bound was achieved up to some constant factor in [LRVW03].

Coming back to the computational analogue, it is natural to study the same question in the domain of pseudorandomness. Given a distribution with dimension s, the problem is to output $O(sn)$ many bits that are pseudorandom. A simple argument can show that deterministic pseudorandom extraction is not possible, but it is not at all clear that how many pure random bits are necessary to serve the purpose. In this thesis, we show that we need to actually involve $\Omega(\log n)$ random bits to extract out $(sn)^\delta$, for any constant $\delta > 0$ many pseudorandom bits from a distribution of dimension at least s. However explicit (or polynomial time computable) construction of one such extractor with $O(\log n)$ random bits is still unknown. If it is known that the given distribution has high HILL-type pseudo min-entropy, then any randomness extractor will work [BSW03]. Instead of HILL-type pseudoentropy, even if we have Yao-type pseudo min-entropy, then also some special kind of randomness extractor (namely with a "reconstruction procedure") could serve our purpose [BSW03]. Unfortunately both of these notions of pseudoentropy can be very small for a distribution with very high dimension. Actually the same counterexample will work for both the cases. So it is interesting to come up with a pseudorandom extractor for a class of distributions having high dimension.

As a first step towards this goal, we consider a special kind of source which we call the *nonpseudorandom bit-fixing source.* It is similar to the well studied notion of *bit-fixing random source* introduced by Chor *et al.* [CGH$^+$85], for which we know the construction of a deterministic randomness extractor due to [KZ03] and [GRS04]. In this thesis, we show that the same construction yields a deterministic pseudorandom extractor for all nonpseudorandom bit-fixing sources.

To the end of Part I of this thesis, we make a little progress towards the question of P vs. BPP by showing that in order to prove $\mathsf{P} = \mathsf{BPP}$, it is sufficient to construct an algorithm that stretches $O(\log n)$ pure random bits to n bits such that the output distribution has a non-zero dimension (not necessarily pseudorandom). The idea is that using such stretching algorithm, we easily construct a hard function, which eventually gives us the most desired optimal pseudorandom generator.

1.1 Organization of Part I of the Thesis

The rest of the Part I of this thesis is organized as follows. In Chapter 2 we describe some common notations used in the Part I of this thesis and also provide a few basic definitions used in the subsequent chapters. In Chapter 3 using the notion of dimension we first define a characterization of distributions based on the amount of pseudorandomness present in them and then in section 3.2, we establish a relationship between dimension and unpredictability of a distribution. In section 3.3 of Chapter 3, we discuss some useful properties of dimension. In Chapter 4 we talk about different known variants of pseudoentropy and connections of them with our notion of dimension. In Section 5.1 of Chapter 5, we introduce the notion of a pseudorandom extractor and provide a lower bound on number of pure random bits required for designing such pseudorandom extractor for distributions with high dimension. Then in Section 5.1.1 we define a special kind of nonpseudorandom source, named nonpseudorandom bit-fixing source and give an explicit construction of a deterministic pseudorandom extractor for such sources. To the end of Chapter 5, in Section 5.2 we show that to completely derandomize polynomial time algorithms we need to design a stretching algorithm that stretches $O(\log n)$ pure random bits exponentially and outputs distribution having non-zero dimension.

Chapter 2

Preliminaries

In this chapter, we give the basic definitions and known theorems used in this thesis. Let us start with the notations used in Part I.

2.1 Notations

We consider the binary alphabet $\Sigma = \{0, 1\}$. We use $D[E]$ to denote $Pr_{x \in_R D}[E]$, where E is an event and x is drawn randomly from the distribution D. We denote the uniform distribution on Σ^m by U_m. Given a string $w \in \Sigma^n$, $w[i]$ denotes the i-th bit of w and $w[1, \ldots, i]$ denotes the first i bits of w. Now suppose $w \in \Sigma^n$ and $I = \{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$ is a set of indices, then we use the notation w_I to denote the string $w[i_1]w[i_2] \ldots w[i_k]$. And we use λ to denote an empty string. For any string $w \in \Sigma^*$, we use the notation $|w|$ to denote the length of the string w and for any $r \in \mathbb{R}$, by $|r|$ we mean the absolute value of r.

For the basic definitions of the computational models like Turing machine model and circuit model and the complexity classes defined on those models used in this thesis, we refer the reader to any standard book on complexity theory (e.g., [AB09]).

2.2 Pseudorandomness

We start by defining the notion of *computational indistinguishability* which we will use frequently in this thesis.

Definition 2.1 (Computational Indistinguishability). *A distribution D over Σ^n is (S, ϵ)-*indistinguishable *from another distribution D' over Σ^n (for $S \in \mathbb{N}, \epsilon > 0$) if for every circuit C of size at most S,*

$$|D[C(x) = 1] - D'[C(x) = 1]| \leq \epsilon.$$

Pseudorandomness is defined via computational indistinguishability [GM84] as well as via unpredictability [Yao82]. It is well-known that both the definitions of pseudorandomness are equivalent [Gol01, AB09]. We state both the definitions below.

Definition 2.2 (Pseudorandomness). *For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$,*[1] *$\epsilon = \epsilon(n) > 0$,*

1. *(via computational indistinguishability) D is said to be (S, ϵ)*-pseudorandom *if for all sufficiently large n, D_n is $(O(S(n)), \epsilon(n))$-indistinguishable from U_n; or equivalently,*

2. *(via unpredictability [Yao82]) D is said to be (S, ϵ)-pseudorandom if for all sufficiently large n,*
$$D_n[C(x_1, \cdots, x_{i-1}) = x_i] \le \frac{1}{2} + \frac{\epsilon(n)}{n}$$
for all circuits C of size at most $O(S(n))$ and for all $i \in [n]$.

It is always natural to consider asymptotic definitions with respect to all polynomial size circuits and allowing bias term to be any inverse polynomial. An ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where a distribution D_n is on Σ^n, is said to be *pseudorandom* if for every constant $c > 0$ and $c' > 0$, D is $(n^c, 1/n^{c'})$-pseudorandom [Gol01].

2.3 Gales and Predictors

Martingales are "fair" betting games which are used extensively in probability theory (see for example, [AL06]). We consider martingale in discrete domain, where it is defined as discrete-time stochastic process that satisfies certain properties (see for example, [Bil95]). In this thesis, we use an alternative definition of martingale defined via betting function [Sch71]. Lutz introduced a generalized notion, that of an s-gale, to characterize Hausdorff dimension [Lut03a]. Athreya *et al.* used a similar notion to characterize packing dimension[AHLM07].

Definition 2.3. *[Lut03a] Let $s \in [0, \infty)$. An s-*gale *is a function $d : \Sigma^* \to [0, \infty)$ such that*
$$d(\lambda) = 1$$
and for all $w \in \Sigma^*$, the following holds,
$$d(w) = 2^{-s}[d(w0) + d(w1)].$$
A martingale *is a 1-gale.*

The following proposition establishes a connection between s-gales and martingales.

Proposition 2.3.1 ([Lut03a]). *A function $d : \Sigma^* \to [0, \infty)$ is an s-gale if and only if the function $d' : \Sigma^* \to [0, \infty)$ defined as $d'(w) = 2^{(1-s)|w|}d(w)$ is a martingale.*

It is already known that the notion of an s-gale is equivalent to the notion of predictors, which have been extensively used in the literature [VZ12]. Given an initial finite segment of a string, a predictor specifies a probability distribution over Σ for the next symbol in the string.

Definition 2.4. *A function $\pi : \Sigma^* \times \Sigma \to [0, 1]$ is a* predictor *if for all $w \in \Sigma^*$,*
$$\pi(w, 0) + \pi(w, 1) = 1.$$

[1]Throughout this thesis, we consider $S(n) > n$ so that the circuit can at least read the full input.

Note: The above definition of a predictor is not much different from the type of predictor used in Definition 2.2. Predictor used in Definition 2.2 actually comes from the predictor that can be simulated by a randomized circuit, by hardwiring "good" random bits. Now if we have a randomized circuit as predictor that given a prefix of a string outputs the next bit, then by invoking that predictor independently polynomially many times we can get an estimate on the probability of occurrence of 0 or 1 as the next bit and using Chernoff bound it can easily be shown that the estimation is correct up to some inverse exponential error. For the detailed equivalence, the reader may refer to [VZ12]. Throughout the thesis, we only consider the martingales (or s-gales) and predictors that can be computed using family of non-uniform circuits and from now onwards we refer them just by martingales (or s-gales) and predictors, and by the size of a martingale (or an s-gale or a predictor), we refer the size of the circuit corresponding to that martingale (or s-gale or predictor).

2.4 Equivalence between s-Gale and Predictor

As we mentioned earlier that there is an equivalence between an s-gale and a predictor. An early reference to this is [Cov74]. Informally, while constructing an s-gale one can invest money to the next symbol proportional to the probability assigned by the predictor. Similarly, if on a particular next bit we get larger amount of money by any betting game then it must be the case that that particular bit will occur with greater probability than the other bit, and thus we predict that bit position accordingly. Here we follow the construction given in [Hit03].

We can define an s-gale d_π for each $s \in [0, \infty)$ from a predictor π as follows:

$$d_\pi(\lambda) = 1$$

and for all $w \in \Sigma^*$, $a \in \Sigma$,

$$d_\pi(wa) = 2^s d_\pi(w)\pi(w, a).$$

Equivalently we can state that for all $w \in \Sigma^*$,

$$d_\pi(w) = 2^{s|w|} \prod_{i=1}^{|w|} \pi(w[1 \cdots i-1], w[i]).$$

Conversely, we can use an s-gale d with $d(\lambda) = 1$ to define a predictor π_d: for all $w \in \Sigma^*$ and $a \in \Sigma$,

$$\pi_d(w, a) = \begin{cases} 2^{-s} \frac{d(wa)}{d(w)} & \text{if } d(w) \neq 0 \\ \frac{1}{2} & \text{otherwise.} \end{cases}$$

In the literature, s-gales have been used to study the dimension of sets of infinite sequences. We refer the reader to [Hita, Hitb] for an extensive bibliography.

2.5 Basics of Information Theory

In this section, we provide a few basic notations and propositions of information theory that we use in the subsequent chapters. For a comprehensive treatment of this topic, we refer the reader to an excellent book by Cover and Thomas [CT06].

Definition 2.5 (Shannon Entropy)**.** *The* Shannon entropy *of a discrete random variable X is defined as*

$$H(X) := -\sum_x Pr[X = x] \log Pr[X = x] = -\mathbb{E}_{x\sim X}[\log Pr[X = x]].$$

The joint entropy $H(X, Y)$ *is defined to be* $-\mathbb{E}_{x\sim X,y\sim Y}[\log Pr[X = x, Y = y]]$ *and the* conditional entropy $H(Y \mid X)$ *is defined to be* $\mathbb{E}_{x\sim X}[H(Y \mid X = x)]$*.*

Proposition 2.5.1 (Chain Rule for Shannon Entropy)**.**

$$H(X, Y) = H(X) + H(Y \mid X).$$

Definition 2.6 (KL divergence)**.** *The* Kullback-Leibler distance *or* KL divergence *between two distributions P and Q is defined as*

$$\mathbf{KL}(P\|Q) := \mathbb{E}_{p\sim P} \log \frac{Pr[P = p]}{Pr[Q = p]}.$$

Definition 2.7 (Conditional KL divergence)**.** *For random variables (P_1, P_2) and (Q_1, Q_2), the* conditional KL divergence *from $(P_2|P_1)$ to $(Q_2|Q_1)$ is defined as*

$$\mathbf{KL}((P_2|P_1)\|(Q_2|Q_1)) = \mathbb{E}_{p_1\sim P_1,p_2\sim P_2}\big[\log \frac{Pr[P_2 = p_2|P_1 = p_1]}{Pr[Q_2 = p_2|Q_1 = p_1]}\big].$$

Just like Shannon entropy, in this case also, we have chain rule stated below.

Proposition 2.5.2 (Chain Rule for KL divergence)**.**

$$\mathbf{KL}(P_1, P_2\|Q_1, Q_2) = \mathbf{KL}(P_1\|Q_1) + \mathbf{KL}((P_2|P_1)\|(Q_2|Q_1)).$$

In the domain of randomness, another important information theoretic entropy measure, namely *min-entropy* is widely used to measure the amount of randomness present in a distribution.

Definition 2.8 (Min-entropy)**.** *For a distribution D,* min-entropy *of D is defined as*

$$H_\infty(D) = \min_x\{\log(1/D[x])\}.$$

Chapter 3

Dimension and Pseudorandomness

In this chapter, we propose a measure to quantify the amount of pseudorandomness present in a distribution. For that purpose, we adapt the notion introduced by Lutz [Lut03b] of an s-gale to define a variant notion of success of an s-gale against a distribution D on Σ^n. We also use the theory on logarithmic loss unpredictability [Hit03] defined on predictor to show a direct equivalence with our proposed notion of dimension. Finally we prove some important properties of our notion of dimension which make our proposed notion robust.

3.1 Quantification of Pseudorandomness

In this section, we propose a quantification of pseudorandomness present in a distribution. Throughout the Part I of this thesis, we will talk about non-uniform definitions.

3.1.1 Defining Dimension

Definition 3.1. *For any $\epsilon > 0$, an s-gale $d : \Sigma^* \to [0, \infty)$ is said to* ϵ-succeed over a distribution D_n *on Σ^n if*

$$D_n[d(w) \geq 2] > \frac{1}{2} + \epsilon.$$

Note that the above definition of win of an s-gale is not arbitrary and reader may refer to the last portion of the proof of Theorem 3.3 to get some intuition behind this definition. The following lemma states the equivalence between the standard definition of pseudorandomness and the definition using martingale.

Lemma 3.1.1. *Consider an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n. If D is (S, ϵ)-pseudorandom then for all large enough n, there is no martingale of size at most $O(S(n))$ that $\epsilon(n)$-succeeds on D_n. Conversely, if for all large enough n, there is no martingale of size at most $O(S(n))$ that $\frac{\epsilon(n)}{n}$-succeeds on D_n, then D is (S, ϵ)-pseudorandom.*

Proof. Assume $d : \Sigma^* \to [0, \infty)$ is a martingale which $\epsilon(n)$-succeeds on D_n for some $n \in \mathbb{N}$, i.e.,

$$D_n[d(w) \geq 2] > \frac{1}{2} + \epsilon(n).$$

Now note that by the definition of martingale the expected value of $d(w)$ over uniform distribution is 1. Hence using Markov Inequality,

$$U_n[d(w) \geq 2] \leq \frac{1}{2}.$$

Let C_d be a circuit of size $O(S(n))$ obtained by instantiating d at length n. Now let C be a circuit which outputs 1 if $C_d(w) \geq 2$. Then,

$$|D_n[C(w) = 1] - U_n[C(w) = 1]| > \epsilon(n).$$

Thus D is not (S, ϵ)-pseudorandom.

Now for the converse direction, assume that D is not (S, ϵ)-pseudorandom. Then there exists an $n_0 \in \mathbb{N}$ such that for any $n \geq n_0$ there exists an bit position $i \in [0, n-1)$ and some circuit C of size at most $O(S(n))$ for which

$$D_n[C(w_1, \cdots, w_{i-1}) = w_i] > \frac{1}{2} + \frac{\epsilon(n)}{n}.$$

Now build a martingale $d : \Sigma^* \to [0, \infty)$ using this circuit C as follows. Let $d(\lambda) = 1$. Now, $\forall j \in [n], j \neq i$, $d(w[0 \ldots j-1]0) = d(w[0 \ldots j-1]1) = d(w[0 \ldots j-1])$, and $d(w[0 \ldots i-1]b) = 2d(w[0 \ldots i-1])$, $d(w[0 \ldots i-1]\bar{b}) = 0$ if $C(w[0 \ldots i-1]) = b$.

Now it is clear that

$$D_n[d(w) \geq 2] > \frac{1}{2} + \frac{\epsilon(n)}{n}$$

and for all large enough n, the size of the martingale d is at most $O(S(n))$. □

The next definition gives a complete quantification of distributions in terms of dimension.

Definition 3.2 (Weak Dimension or Dimension)**.** *For an ensemble of distributions $D = \{D_n\}_{n \in \mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, the (S, ϵ)-*weak dimension *or simply* dimension *of D is defined as*

$$\dim_{S,\epsilon}(D) = \inf\{s \in [0, \infty) \mid \textit{for infinitely many } n, \exists s\textit{-gale } d \textit{ of size at most } O(S(n)) \textit{ which } \epsilon(n)\textit{-succeeds on } D_n\}.$$

We refer the reader to Section 3.3 to find the justification behind the fact that the above definition of dimension is well-defined. Informally, if the dimension of an ensemble of distribution is s, we say that it is *s-pseudorandom*. It is clear from Lemma 3.1.1 that for any (S, ϵ)-pseudorandom ensemble of distributions D, $\dim_{S,\epsilon}(D) \geq 1$. In Section 3.3 we will see that it is actually an equality. Reader may find examples of s-pseudorandom distribution for arbitrary $s \in [0, 1]$, in the proof of Theorem 3.3. We can also define dimension of distributions in slightly stronger sense.

Definition 3.3 (Strong Dimension)**.** *For an ensemble of distributions $D = \{D_n\}_{n \in \mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, the (S, ϵ)-*

strong dimension *of D is defined as*

$$\mathrm{sdim}_{S,\epsilon}(D) = \inf\{s \in [0,\infty) \mid \textit{for all large enough } n, \exists s\textit{-gale } d \textit{ of size at most } O(S(n)) \textit{ which } \epsilon(n)\textit{-succeeds on } D_n\}.$$

It follows from the definition that weak dimension is smaller or equal to strong dimension.

Proposition 3.1.1. *For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, $\dim_{S,\epsilon}(D)$ and $\mathrm{sdim}_{S,\epsilon}(D)$ are well defined and*

$$\dim_{S,\epsilon}(D) \leq \mathrm{sdim}_{S,\epsilon}(D).$$

We will show separation between both of these notions of dimensions in Section 3.3. Now just like the asymptotic definition of pseudorandomness with respect to all polynomial size circuits and inverse polynomial bias, for any ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, we can give definitions of weak dimension or simply dimension (denoted by $\dim(D)$) and strong dimension (denoted by $\mathrm{sdim}(D)$) by allowing $S(n)$ to be $n^{O(1)}$ and $\epsilon(n)$ to be $n^{-O(1)}$ in the Definition 3.2 and 3.3 respectively.

3.2 Unpredictability and Dimension

It is customary to measure the performance of a predictor utilizing a *loss function* [Hit04]. The loss function determines the penalty incurred by a predictor for erring in its prediction. Let the next bit be b and the probability induced by the predictor on it is p_b.

Commonly used loss functions include the *absolute loss function*, which penalizes the amount $1 - p_b$; and the *logarithmic loss function*, which penalizes $-\log(p_b)$. The latter, which appears complicated at first glance, is intimately related to the concepts of Shannon Entropy and dimension. In this section, adapting the result of Hitchcock [Hit03], we establish that there is an equivalence between the notion of dimension that we have defined in the previous section, and the logarithmic loss function defined on a predictor.

Definition 3.4. *The* logarithmic loss function *on $p \in [0,1]$ is defined to be $loss(p) = -\log p$.*

Using this, we define the running loss that a predictor incurs while it predicts successive bits of a string in Σ^n, as the sum of the losses that the predictor makes on individual bits.

Definition 3.5. *Let $\pi : \Sigma^* \times \Sigma \to [0,1]$ be a predictor.*

1. *The* cumulative loss *of π on $w \in \Sigma^n$, denoted as $Loss(\pi, w)$, is defined by $Loss(\pi, w) = \sum_{i=1}^{n} loss(\pi(w[1 \ldots i-1],\ w[i]))$.*

2. *The* loss rate *of π on $w \in \Sigma^n$ is $LossRate(\pi, w) = \frac{Loss(\pi,w)}{n}$.*

3. *The* ϵ-loss rate *of π over a distribution D_n on Σ^n is*

$$LossRate_\epsilon(\pi, D_n) = \inf\{t \in [0,1] \mid D_n[LossRate(\pi, w) \leq t] > \frac{1}{2} + \epsilon\}.$$

Intuitively the unpredictability of a distribution is defined as the infimum of the loss rate that any predictor has to incur on the distribution.

Definition 3.6 (Weak Unpredictability or Unpredictability). *For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, the (S, ϵ)-*weak unpredictability *or simply* unpredictability *of D is*

$$\text{unpred}_{S,\epsilon}(D) = \inf\{t \in [0,1] \mid \textit{for infinitely many } n\textit{, there exists a predictor } \pi \textit{ of size at most } O(S(n)) \textit{ such that } \textit{LossRate}_{\epsilon(n)}(\pi, D_n) \le t\}.$$

With this, we can prove that dimension can equivalently be defined using unpredictability. The proof is motivated from the proof of the equivalence between logarithmic loss unpredictability and dimension [Hit03].

Theorem 3.1. *Consider any $s \in [0,1]$. For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, if* $\dim_{S,\epsilon}(D) \le s$ *then* $\text{unpred}_{S^{O(1)},\epsilon}(D) \le s$*. Conversely,* $\text{unpred}_{S,\epsilon}(D) \le s$ *implies* $\dim_{S^{O(1)},\epsilon}(D) \le s$.

Proof. Assume that s' is any number such that $s < s'$ and then take a number s'' such that $s < s'' \le s'$ and $2^{s''}$ is rational[1]. For some large enough n, suppose d is an s''-gale of size at most $O(S(n))$ that $\epsilon(n)$-succeeds on D_n. Let $\pi_d : \Sigma^* \times \Sigma \to [0,1]$ be defined by

$$\pi_d(w,b) = \begin{cases} 2^{-s''}\frac{d(wb)}{d(w)} & \text{if } d(w) \neq 0 \\ \frac{1}{2} & \textit{otherwise.} \end{cases}$$

For any $w \in \Sigma^n$ with $d(w) \ge 2$, we have

$$\begin{aligned} \text{Loss}_{\pi_d}(w) &= -\sum_{i=1}^{n} \log \pi_d(w[1\ldots i-1], w[i]) \\ &= -\log \Pi_{i=1}^{n} \pi_d(w[1\ldots i-1], w[i]) \\ &= s''n - \log d(w) \\ &\le s''n - 1 \le s'n. \end{aligned}$$

So $\text{LossRate}(\pi_d, w) \le s'$. Thus,

$$D_n[\text{LossRate}(\pi_d, w) \le s'] \ge D_n[d(w) \ge 2] > \frac{1}{2} + \epsilon.$$

Note that implementation of π_d involves division of two at most $O(S(n))$ bits rational numbers[2] and thus can be done using a circuit of size at most $(S(n))^{O(1)}$ [Vol99].

Conversely, assume that $\text{unpred}_{S,\epsilon}(D) \le t \in [0,1]$. Assume that t' is any number satisfying $t < t'$ and then take any number t'' such that $t' < t''$ and $2^{t''}$ is rational. For

[1]We consider $2^{s''}$ to be rational to ensure that the value of $2^{s''}$ can be computed using constant size circuit. Note that we can always find such s'' due to the fact that the function 2^s for $s > 0$ is continuous, monotonically increasing and within any two real numbers there exists a rational number.

[2]As we are considering martingales and predictors that can be implemented by circuits of size $O(S(n))$ so the output must be a rational number which can be represented by at most $O(S(n))$ bits.

some large enough n, let π be a predictor of size at most $O(S(n))$ such that

$$D_n[\text{LossRate}(\pi, w) \le t'] > \frac{1}{2} + \epsilon.$$

If d_π is the t''-gale defined by

$$d_\pi(w) = 2^{t''|w|}\Pi_{i=1}^{|w|}\pi(w[0 \dots i-1], w[i])$$

then for any $w \in \Sigma^n$ with $\text{LossRate}(\pi, w) \le t'$, we have the following,

$$\begin{aligned}\log d_\pi(w) &= t''n + \sum_{i=1}^{n} \log \pi(w[1 \dots i-1], w[i]) \\ &= t''n - \text{Loss}_\pi(w) \ge 1.\end{aligned}$$

The last inequality holds for all sufficiently large n. Hence, for infinitely many n,

$$D_n[d_\pi(w) \ge 2] > \frac{1}{2} + \epsilon.$$

Moreover, computation of d involves multiplication of n rational numbers of at most $O(S(n))$ bits each and thus can be implemented by a circuit of size $(S(n))^{O(1)}$ [Vol99]. □

Just like dimension, one can also define strong unpredictability in the following way.

Definition 3.7 (Strong Unpredictability)**.** *For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, the (S, ϵ)-*strong unpredictability *of D is*

$$\text{sunpred}_{S,\epsilon}(D) = \inf\{t \in [0, 1] \mid \textit{for all large enough } n\textit{, there exists a predictor } \pi \textit{ of size at most } O(S(n)) \textit{ such that } \textit{LossRate}_{\epsilon(n)}(\pi, D_n) \le t\}.$$

Now by following the proof of Theorem 3.1 it is easy to see that a similar relation also holds between strong dimension and strong unpredictability.

Theorem 3.2. *Consider any $s \in [0, 1]$. For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, if* $\text{sdim}_{S,\epsilon}(D) \le s$ *then* $\text{sunpred}_{S^{O(1)},\epsilon}(D) \le s$*. Conversely,* $\text{sunpred}_{S,\epsilon}(D) \le s$ *implies* $\text{sdim}_{S^{O(1)},\epsilon}(D) \le s$*.*

Analogous to pseudorandomness and dimension, for any ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, one can define weak unpredictability or simply unpredictability (denoted by $\text{unpred}(D)$) and strong unpredictability (denoted by $\text{sunpred}(D)$) by allowing $S(n)$ to be $n^{O(1)}$ and $\epsilon(n)$ to be $n^{-O(1)}$ in the Definition 3.6 and 3.7 respectively. Following is a straight forward implication of Theorem 3.1 and Theorem 3.2.

Corollary 3.2.1. *For any ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n,*

$$\text{dim}(D) = \text{unpred}(D) \textit{ and } \text{sdim}(D) = \text{sunpred}(D).$$

3.3 Properties of Dimension

We now establish a few basic properties of our notion of dimension. We begin by exhibiting existence of an ensemble of distributions with dimension s, for any $s \in [0,1]$.

First, we observe that the dimension of any ensemble of distributions D is the infimum of a non-empty subset of $[0, 1+\epsilon]$ for any $\epsilon > 0$ and hence the dimension of a distribution is well-defined. The following lemma establishes the above claim.

Lemma 3.3.1. *For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, $\mathrm{sdim}_{S,\epsilon}(D) \leq 1$.*

Proof. Let us first take any $s > 1$ such that 2^s is rational and then consider the following function $d : \Sigma^* \to [0,\infty)$. Let $d(\lambda) = 1$ and $\forall i \in [n]$, $d(w[0\ldots i-1]0) = d(w[0\ldots i-1]1) = 2^{s-1}d(w[0\ldots i-1])$. It is easy to see that d is an s-gale and for any $w \in \Sigma^n$, $d(w) = 2^{(s-1)n}$. Thus for all large enough n, d will $\epsilon(n)$-succeed over D_n. Also note that for all large enough n, this function d can be implemented using a circuit of size $O(n)$.[3] Hence the statement of the lemma follows. □

Above lemma along with Proposition 3.1.1 implies the following corollary.

Corollary 3.3.1. *For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, $\dim_{S,\epsilon}(D) \leq 1$.*

Since it is clear that any ensemble of distributions has a dimension, the following theorem establishes the fact that our definition yields a nontrivial quantification of the set of ensembles of distributions.

Theorem 3.3. *Let $s \in [0,1]$. Then for for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, there is an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where the distribution D_n is on Σ^n, such that (S,ϵ)-dimension (also strong dimension) of D is s.*

Proof. Let us take the ensemble of uniform distributions, i.e., $D := \{U_n\}_{n\in\mathbb{N}}$, where U_n is the uniform distribution on Σ^n. Then by Lemma 3.1.1 together with Lemma 3.3.1 shows that for any S and ϵ, $\mathrm{sdim}_{S,\epsilon}(D) = 1$. By using corollary 3.3.1 instead of Lemma 3.3.1, one can also show that $\dim_{S,\epsilon}(D) = 1$.

On the other hand we consider an ensemble $D = \{D_n\}_{n\in\mathbb{N}}$ where support size of each D_n is one, or in other words, D_n imposes all the probability on a single string, say 0^n. Now first take any $s > 0$ such that 2^s is rational and then consider the following function $d : \Sigma^* \to [0,\infty)$. Let $d(\lambda) = 1$ and $\forall i \in [n]$, $d(w[0\ldots i-1]0) = 2^s d(w[0\ldots i-1])$. It is easy to see that d is an s-gale and $d(0^n) = 2^{sn}$. Thus for all large enough n, d will $\epsilon(n)$-succeed over D_n. Also note that for all large enough n this function d can be implemented using a circuit of size $O(n)$. Hence for any S and ϵ, (S,ϵ)-dimension as well as (S,ϵ)-strong dimension of D is 0.

Otherwise, assume that $s \in (0,1)$. Let us take the ensemble of uniform distributions $D := \{U_n\}_{n\in\mathbb{N}}$. To each string $x \in \Sigma^n$, we append $\lfloor \frac{n}{s} \rfloor - n$ many zeros, and denote the resulting string as x'. Let $D'_n(x') = U_n(x)$. For strings $y \in \Sigma^{\lfloor \frac{n}{s} \rfloor}$ which do not terminate in a sequence of $\lfloor \frac{n}{s} \rfloor - n$ many zeros, we set $D'_n(y) = 0$.

[3] As for every string w of length n, the value of $d(w)$ will be same, so one can hardcode the value $2^{(s-1)n}$ inside the non-uniform circuit implementing the function d.

For any large enough n, let $\pi : \Sigma^* \times \Sigma \to [0,1]$ be the predictor for distribution U_n which testifies that the $(S^{O(1)}, \epsilon)$-strong unpredictability of D is at most 1. Define the new predictor $\pi' : \Sigma^* \times \Sigma \to [0,1]$ by

$$\pi'(x,b) = \begin{cases} \pi(x,b) & \text{if}|x| < n, b = 0,1 \\ 1 & \text{if}|x| \geq n, b = 0 \\ 0 & \text{otherwise.} \end{cases}$$

For every $w \in \Sigma^{\lfloor \frac{n}{s} \rfloor}$ which is in the support of D'_n such that $\text{LossRate}(\pi, w[1 \dots n]) \leq (1+\epsilon_1)$, for any $\epsilon_1 > 0$, we have that

$$\text{LossRate}(\pi', w) = \frac{\text{Loss}(\pi, w[1 \dots n])}{\lfloor \frac{n}{s} \rfloor} \leq \frac{(1+\epsilon_1)n}{\lfloor \frac{n}{s} \rfloor} \leq (s + \epsilon'), \quad \text{for some } \epsilon' > 0$$

The last inequality holds for all small enough s/n and this testifies that the $(S^{O(1)}, \epsilon)$-strong unpredictability (hence the $(S^{O(1)}, \epsilon)$-strong dimension) of the ensemble of distributions D' is at most s. Now by Proposition 3.1.1, it follows that $\dim_{S^{O(1)},\epsilon}(D')$ is also at most s.

Now, assume that (S, ϵ)-dimension of D' is less than s. For any s' such that $0 < s' < s$, for infinitely many n, there exists an s'-gale d of size at most $O(S(n))$ which $\epsilon(n)$-succeeds on D'_n. We show that this would imply that U_n is not uniform. Now consider a string $w \in \Sigma^{\lfloor \frac{n}{s} \rfloor}$, which is in the support of D'_n. For any $k \in \{n+1, \cdots, \lfloor \frac{n}{s} \rfloor\}$, $d(w[1 \dots k]) \leq 2^{s'} d(w[1 \dots k-1])$ and thus $d(w) \geq 2$ will imply that $d(w[1 \dots n]) \geq 2^{-s'(\lfloor \frac{n}{s} \rfloor - n)+1}$. Now consider the martingale d' (needs not be computed by any circuit) corresponding to the s'-gale d. According to [Lut03a], we have $d'(w') = 2^{(1-s')|w'|} d(w')$, for any string $w' \in \Sigma^*$. Thus,

$$\begin{aligned} D'_n[d'(w[1 \dots n]) \geq 2] &\geq D'_n[d(w[1 \dots n]) \geq 2^{-s'(\lfloor \frac{n}{s} \rfloor - n)+1}] \\ &\geq D'_n[d(w) \geq 2] \\ &> \frac{1}{2} + \epsilon(n). \end{aligned}$$

Note that $D'_n[d'(w[1 \dots n]) \geq 2]$ is same as $U_n[d'(x) \geq 2]$, which contradicts the fact that by Markov inequality, $U_n[d'(x) \geq 2] \leq \frac{1}{2}$. This shows that $\dim_{S,\epsilon}(D') \geq s$ and hence by Proposition 3.1.1, $\text{sdim}_{S,\epsilon}(D')$ is also greater than or equal to s. $\square$

Informally the following theorem shows that our notion of dimension is able to capture the fact that if we mix a "good" distribution with a small amount of an extremely "bad" distribution, then also "quality" of the first distribution would not change much.

Theorem 3.4. *Let $D = \{D_n\}_{n \in \mathbb{N}}$, $D^1 = \{D^1_n\}_{n \in \mathbb{N}}$ and $D^2 = \{D^2_n\}_{n \in \mathbb{N}}$ be three ensembles of distributions such that for some $\delta = \delta(n) \in [0,1]$, for every $n \in \mathbb{N}$, $D_n = (1-\delta(n))D^1_n + \delta(n)D^2_n$. If for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, $\dim_{S,\epsilon}(D^1) = s_1$ ($\text{sdim}_{S,\epsilon}(D^1) = s_1$), then $\dim_{S,(\epsilon+\delta)}(D) \geq s_1$ ($\text{sdim}_{S,(\epsilon+\delta)}(D) \geq s_1$).*[4]

Proof. For the contrary, let us assume that, $\dim_{S,(\epsilon+\delta)}(D) < s_1$ and assume $s = s_1 - \epsilon_1$, for some ϵ_1, $0 < \epsilon_1 < s_1$. So for infinitely many n, there exists an s-gale of size at most

[4]Note that bias term in the dimension of D^1 depends on δ.

$O(S(n))$ that $(\epsilon(n) + \delta(n))$-succeeds over D_n. Thus,

$$D_n[d(w) \geq 2] > \frac{1}{2} + (\epsilon(n) + \delta(n)).$$

Let the strings w for which $d(w) \geq 2$ holds be w_i, $1 \leq i \leq k$. So,

$$D_n(w_1) + \cdots + D_n(w_k) > \frac{1}{2} + (\epsilon(n) + \delta(n)),$$

where $D_n(w_i) = (1 - \delta(n))D_n^1(w_i) + \delta(n)D_n^2(w_i)$, for all $1 \leq i \leq k$.
Now, as $D_n^2(w_1) + \cdots + D_n^2(w_k) \leq 1$,

$$D_n^1(w_1) + \cdots + D_n^1(w_k) > \frac{1}{2} + \epsilon(n)$$

and thus $\dim_{S,\epsilon}(D^1) < s_1$ which is a contradiction.

The above argument is valid for all n for which there exists an s-gale of size at most $O(S(n))$ that $(\epsilon(n) + \delta(n))$-succeeds over D_n and hence the same claim holds even if we consider strong dimension instead of dimension. $\square$

If we follow the proof of Theorem 3.4 with martingale instead of s-gale, we get the following weaker version of the above theorem, which we will require in the construction of deterministic extractor for a special kind of sources in Section 5.1.1.

Lemma 3.3.2. *Let $D = \{D_n\}_{n\in\mathbb{N}}$, $D^1 = \{D_n^1\}_{n\in\mathbb{N}}$ and $D^2 = \{D_n^2\}_{n\in\mathbb{N}}$ be three ensembles of distributions such that for some $\delta = \delta(n) \in [0,1]$, for every $n \in \mathbb{N}$, $D_n = (1-\delta(n))D_n^1 + \delta(n)D_n^2$. If for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, D^1 is (S,ϵ)-pseudorandom, then D is $(S, \epsilon + \delta)$-pseudorandom.*

In subsequent sections, we will see how to extract pseudorandom parts from a convex combination of distributions. For that purpose we need the following lemma which establishes the fact that convex combination of two pseudorandom distributions will be pseudorandom as well.

Lemma 3.3.3. *Consider two ensembles of distributions $D^1 = \{D_n^1\}_{n\in\mathbb{N}}$ and $D^2 = \{D_n^2\}_{n\in\mathbb{N}}$. Suppose for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, both D^1 and D^2 are (S,ϵ)-pseudorandom. If for $\delta = \delta(n) \in [0,1]$ there exists an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$ which can be expressed as for all $n \in \mathbb{N}$, $D_n = \delta(n)D_n^1 + (1-\delta(n))D_n^2$, then D is also (S,ϵ')-pseudorandom, where $\epsilon'(n) = n \cdot \epsilon(n)$.*[5]

Proof. The claim clearly holds when $\delta(n)$ is either 0 or 1, so assume that $0 < \delta(n) < 1$. For the contrary, let us assume that D is not (S,ϵ')-pseudorandom where $\epsilon'(n) = n \cdot \epsilon(n)$, i.e., by Lemma 3.1.1, for infinitely many $n \in \mathbb{N}$ there exists an martingale d of size at most $O(S(n))$ that $\epsilon(n)$-succeeds on D_n, i.e.,

$$D_n[d(w) \geq 2] > \frac{1}{2} + \epsilon(n).$$

As D^2 is (S,ϵ)-pseudorandom, by Lemma 3.1.1, for all large enough n there exists no martingale of size at most $O(S(n))$ that $\epsilon(n)$-succeeds on D_n^2. Thus it is possible to

[5]Note that this lemma will be useful only when we consider $\epsilon(n) < \frac{1}{2n}$.

consider an $n \in \mathbb{N}$ such that there exists a martingale d of size at most $O(S(n))$ that $\epsilon(n)$-succeeds on D_n, but does not $\epsilon(n)$-succeed on D^2_n. Let the strings $w \in \Sigma^n$ for which $d(w) \geq 2$ holds be w_i, $1 \leq i \leq k$. So,

$$D_n(w_1) + \cdots + D_n(w_k) > \frac{1}{2} + \epsilon(n),$$

where $D_n(w_i) = \delta(n)D^1_n(w_i) + (1 - \delta(n))D^2_n(w_i), 1 \leq i \leq k$. Now, since we have that

$$D^2_n(w_1) + \cdots + D^2_n(w_k) \leq \frac{1}{2} + \epsilon(n).$$

Thus the following holds

$$D^1_n(w_1) + \cdots + D^1_n(w_k) > \frac{1}{2} + \epsilon(n).$$

As for infinitely many $n \in \mathbb{N}$ the above happens, so D^1 is not (S, ϵ)-pseudorandom, which is a contradiction. □

We can also slightly generalize the above lemma and get the following theorem.

Theorem 3.5. *Let $D = \{D_n\}_{n\in\mathbb{N}}$, $D^1 = \{D^1_n\}_{n\in\mathbb{N}}$ and $D^2 = \{D^2_n\}_{n\in\mathbb{N}}$ be three ensembles of distributions such that for some $\delta = \delta(n) \in [0, 1]$, for every $n \in \mathbb{N}$, $D_n = \delta(n)D^1_n + (1 - \delta(n))D^2_n$. Then for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$,*

$$\dim_{S,\epsilon}(D) \geq \min\{\dim_{S,\epsilon}(D^1), \dim_{S,\epsilon}(D^2)\}.$$

Proof. The claim clearly holds when $\delta(n)$ is either 0 or 1, so assume that $0 < \delta(n) < 1$. Let $\dim_{S,\epsilon}(D_1) = s_1$, and $\dim_{S,\epsilon}(D_2) = s_2$.

For the contrary, let us assume that, $\dim_{S,\epsilon}(D) < \min\{s_1, s_2\}$. Now consider $s = \min\{s_1, s_2\} - \epsilon_1$, for some $\epsilon_1, 0 < \epsilon_1 < \min\{s_1, s_2\}$. Then for infinitely many n, there exists an s-gale d of size at most $O(S(n))$ such that

$$D_n[d(w) \geq 2] > \frac{1}{2} + \epsilon(n).$$

As $\dim_{S,\epsilon}(D^2) = s_2$, so for all large enough n, there exists no s-gale of size at most $O(S(n))$ that $\epsilon(n)$-succeeds on D^2_n. Thus we can consider an n such that there exists an s-gale d of size at most $O(S(n))$ that $\epsilon(n)$-succeeds on D_n, but does not $\epsilon(n)$-succeed on D^2_n. Let the strings w for which $d(w) \geq 2$ holds be w_i, $1 \leq i \leq k$. So,

$$D_n(w_1) + \cdots + D_n(w_k) > \frac{1}{2} + \epsilon(n)$$

where $D_n(w_i) = \delta(n)D^1_n(w_i) + (1 - \delta(n))D^2_n(w_i)$, for $1 \leq i \leq k$. Also, we have that

$$D^2_n(w_1) + \cdots + D^2_n(w_k) \leq \frac{1}{2} + \epsilon(n).$$

Thus

$$D^1_n(w_1) + \cdots + D^1_n(w_k) > \frac{1}{2} + \epsilon(n)$$

and this happens for infinitely many $n \in \mathbb{N}$ implying $\dim_{S,\epsilon}(D^1) < s_1$, which is a contradiction. $\square$

The following theorem shows that in order for a distribution to have dimension less than 1, it is not sufficient to have a few positions where we can successfully predict - it is necessary that these positions occur often.

Theorem 3.6. *For any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, there is an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$ such that $\dim_{S,\epsilon}(D) = 1$, but D is not (S, ϵ)-pseudorandom.*

Proof. Let D_n on Σ^n be defined as follows.

$$D_n(x) = \begin{cases} \frac{1}{2^{n-1}} & \text{if } x[n] = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then D is clearly not (S, ϵ)-pseudorandom, for any value of $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$. Consider a predictor $\pi : \Sigma^* \times \Sigma \to [0, 1]$ defined as follows. For strings w of length i, $i \in [1, n-2]$, set $\pi(w, b) = 0.5$, $b = 0, 1$ and $\pi(w, 0) = 1$, $\pi(w, 1) = 0$ otherwise. Then

$$D_n[\pi(x[\ 1 \dots n-1\ ], x[n]) = 1] = 1.$$

However, we will show that $\dim_{S,\epsilon}(D) = 1$. Assume that $\dim_{S,\epsilon}(D) < 1$ and thus for some ϵ_1, $0 < \epsilon_1 < 1$, for infinitely many $n \in \mathbb{N}$, there exists an s-gale d, where $s = 1 - \epsilon_1$, of size at most $O(S(n))$ which $\epsilon(n)$-succeeds on D_n. Now consider a string $w \in \Sigma^n$, which is in the support of D_n. Now, $d(w) \le 2^s d(w[1 \dots n-1])$ and thus $d(w) \ge 2$ will imply that $d(w[1 \dots n-1]) \ge 2^{1-s}$. Next consider the martingale d' (needs not be computed by any circuit) corresponding to the s-gale d. According to [Lut03a], we have $d'(w') = 2^{(1-s)|w'|}d(w')$, for any string $w' \in \Sigma^*$. Thus,

$$\begin{aligned} D_n[d'(w[1 \dots n-1]) \ge 2] &\ge D_n[d(w[1 \dots n-1]) \ge 2^{1-s}] \\ &\ge D_n[d(w) \ge 2] \\ &> \frac{1}{2} + \epsilon(n). \end{aligned}$$

The first inequality holds for all large enough n. Note that $D_n[d'(w[1 \dots n-1]) \ge 2]$ is same as $U_{n-1}[d'(x) \ge 2]$, where $x \in \Sigma^{n-1}$ is drawn according to the distribution U_{n-1}. However by Markov inequality, $U_{n-1}[d'(x) \ge 2] \le \frac{1}{2}$, which is a contradiction and this completes the proof. $\square$

We now give separation between two notions of dimension by providing an example of ensemble which has a very high strong dimension, but very low weak dimension.

Theorem 3.7. *There exists an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$ such that for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, $\dim_{S,\epsilon}(D) < \mathrm{sdim}_{S,\epsilon}(D)$.*

Proof. Here we actually show a much stronger claim by giving an example of an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$ such that $\mathrm{sdim}_{S,\epsilon}(D) = 1$ whereas $\dim_{S,\epsilon}(D) = 0$. This is the largest possible gap between weak and strong dimension of any ensemble of distributions.

Construct an ensemble of distributions $D = \{D_n\}_{n \in \mathbb{N}}$ as follows: for all the odd value of n, set $D_n = U_n$ where U_n is the uniform distribution over Σ^n and for all even value of n, set D_n to be such that it imposes all the probability on a single string 0^n.

Due to Markov inequality, there exists no martingale that $\epsilon(n)$-succeeds over distribution D_n for any odd value of n and by Lemma 3.3.1, strong dimension can be at most 1. Hence, $\mathrm{sdim}_{S,\epsilon}(D) = 1$. By the argument used in the proof of Theorem 3.3, we can say that for any $s > 0$ such that 2^s is rational, for all large enough even value of n, there exists an s-gale of size at most $O(S(n))$ that $\epsilon(n)$-succeeds over D_n and hence $\dim_{S,\epsilon}(D) = 0$. $\square$

Chapter 4

Pseudoentropy and Dimension

In this chapter we study the relation between our notion of dimension and different variants of computational or pseudo (min/Shannon) entropy. We first give a brief overview of different notions of pseudoentropy and then establish relations between our notion of dimension and two variants of pseudoentropy.

4.1 Different Notions of Pseudoentropy

Let us first recall that for a distribution D, the *min-entropy* of D is defined as $H_\infty(D) = \min_w\{\log(1/D[w])\}$. We start with the standard definition of computational min-entropy, as given by [HILL99].

Definition 4.1 (HILL-type pseudo min-entropy). *For any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$ has (S,ϵ)-*HILL-type pseudo min-entropy *(or simply (S,ϵ)-pseudo min-entropy) at least $k = k(n)$, denoted as $H_\infty^{HILL,S,\epsilon}(D) \geq k$ if there exists an ensemble of distributions $D' = \{D'_n\}_{n\in\mathbb{N}}$ such that for all large enough n,*

1. *$H_\infty(D'_n) \geq k(n)$, and*
2. *D'_n is $(O(S(n)), \epsilon(n))$-indistinguishable from the distribution D_n.*

Another type of pseudo min-entropy has been studied in the literature where the order of the quantifier of the above definition is reversed and is known as *metric-type pseudo min-entropy.*

Definition 4.2 (Metric-type pseudo min-entropy). *For any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$ has (S,ϵ)-*Metric-type pseudo min-entropy *at least $k = k(n)$, denoted as $H_\infty^{metric,S,\epsilon}(D) \geq k$ if for every family of non-uniform circuits $C = \{C_n\}_{n\in\mathbb{N}}$, where each C_n is of size at most $O(S(n))$, there exists an ensemble of distributions $D' = \{D'_n\}_{n\in\mathbb{N}}$ such that for all large enough n,*

1. *$H_\infty(D'_n) \geq k(n)$, and*
2. *$|D_n[C_n(x) = 1] - D'_n[C_n(x) = 1]| \leq \epsilon(n)$.*

From the definition it is clear that for every ensemble of distributions D, $H_\infty^{metric,S,\epsilon}(D) \geq H_\infty^{HILL,S,\epsilon}(D)$. Converse is also true by the following result from [BSW03] up to some small losses in ϵ and size of the circuit.

Theorem 4.1 ([BSW03]). *For any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, $\delta = \delta(n) > 0$ and $k = k(n)$, for an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, if $H_\infty^{metric,S,(\epsilon-\delta)}(D) \geq k$, then $H_\infty^{HILL,S',\epsilon}(D) \geq k$, where $S'(n) = \Theta(\frac{\delta(n)^2 S(n)}{n})$.*

Due to the above stated result it is sufficient to study the connections among HILL-type pseudo min-entropy and our notion of dimension. Any relation between them directly gives us relation between metric-type pseudo min-entropy and dimension.

Before proceeding further, we want to mention that another type of pseudo min-entropy, namely Yao-type or compression type is present in the literature. However, in this thesis we do not consider it. We refer the reader to [BSW03] for further exposition on this topic.

So far we have talked about pseudo min-entropy. In similar fashion one can also define *pseudo Shannon entropy* and a natural generalization of it is *conditional pseudo Shannon entropy* [HLR07, HRV10, VZ12].

Definition 4.3 (Conditional pseudo Shannon entropy). *A random variable Y_n jointly distributed with X_n is said to have (S,ϵ)-*conditional pseudo Shannon entropy *at least k, for any $S \in \mathbb{N}$ and $\epsilon > 0$ if there exists a random variable Z_n jointly distributed with X_n such that*

1. *$H(Z_n|X_n) \geq k$, and*
2. *(X_n, Y_n) and (X_n, Z_n) are (S,ϵ)-indistinguishable.*

*Now suppose $Y = \{Y_n\}_{n\in\mathbb{N}}$ is an ensemble of random variables jointly distributed with another ensemble of distributions $X = \{X_n\}_{n\in\mathbb{N}}$. For any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, Y is said to have (S,ϵ)-*conditional pseudo Shannon entropy *at least $k = k(n)$ given X if there exists another ensemble of distributions Z jointly distributed with X such that for all sufficiently large n, Y_n has $(O(S(n)),\epsilon(n))$-conditional pseudo Shannon entropy at least $k(n)$.*

The following is another variant of pseudoentropy which was introduced by Haitner, Reingold and Vadhan [HRV10] while giving efficient construction of pseudorandom generator from any one-way function.

Definition 4.4 (Next-bit pseudo Shannon entropy). *An ensemble of random variables $X = \{X_n\}_{n\in\mathbb{N}}$, where $X_n = (X_n^1, X_n^2, \cdots, X_n^n)$ takes values in Σ^n, has (S,ϵ)-*next-bit pseudo Shannon entropy *for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$ at least $k = k(n)$, denoted as $H^{next,S,\epsilon}(X) \geq k$ if there exists an ensemble of random variables $Y = \{Y_n\}_{n\in\mathbb{N}}$ where $Y_n = (Y_n^1, Y_n^2, \cdots, Y_n^n)$ takes values in Σ^n and for all sufficiently large n,*

1. *$\sum_i H(Y_n^i|X_n^1, \cdots, X_n^{i-1}) \geq k(n)$, and*
2. *for all $1 \leq i \leq n$, $(X_n^1, \cdots, X_n^{i-1}, X_n^i)$ and $(X_n^1, \cdots, X_n^{i-1}, Y_n^i)$ are $(O(S(n)),\epsilon(n))$-indistinguishable.*

4.2 High HILL-type pseudo min-entropy implies high dimension

In this section we do a comparative study between HILL-type pseudo min-entropy and our notion of dimension.

Theorem 4.2. *For every ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$ and for any $S = S(n) > n$, $\epsilon = \epsilon(n) > 0$, if $H_\infty^{HILL,S,\epsilon}(D) \geq k$, where $k = k(n) = sn$ for some $s \in [0,1]$, then $\dim_{S,\epsilon}(D) \geq s$.*

Proof. The theorem is a consequence of the following lemma.

Lemma 4.2.1. *For every ensemble of distributions $X = \{X_n\}_{n\in\mathbb{N}}$, if for all large enough n, $H_\infty(X_n) \geq k(n)$, where $k(n) = sn$ for some $s \in [0,1]$, then $\dim_{S,\epsilon}(X) \geq s$ for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$.*

Now observe that if for all large enough n, distribution D_n is $(O(S(n)), \epsilon(n))$-indistinguishable from another distribution D'_n, then $\dim_{S,\epsilon}(D) = \dim_{S,\epsilon}(D')$ as otherwise the s-gale of size at most $O(S(n))$ which $\epsilon(n)$-succeeds over exactly one of them, acts as a distinguishing circuit. This fact along with Lemma 4.2.1 completes the proof. □

It only remains to establish Lemma 4.2.1.

Proof of Lemma 4.2.1. For the sake of contradiction, let us assume that for infinitely many $n \in \mathbb{N}$, there exists an s-gale d of size at most $O(S(n))$ that $\epsilon(n)$-succeeds over X_n, i.e.,

$$X_n[d(w) \geq 2] > \frac{1}{2} + \epsilon(n).$$

Now consider the following set

$$S := \{w \in \Sigma^n \mid d(w) \geq 2\}.$$

As $H_\infty(X_n) \geq k(n)$,

$$|S| > 2^{sn-1} + 2^{sn} \cdot \epsilon(n).$$

By taking the corresponding martingale d' (needs not be computed by any circuit) according to the Proposition 2.3.1, we have that for any $w \in S$, $d'(w) \geq 2^{(1-s)n+1}$ and as a consequence,

$$U_n[d'(w) \geq 2^{(1-s)n+1}] > 2^{sn-n-1} + 2^{sn-n} \cdot \epsilon(n)$$

which contradicts the fact that by Markov inequality, $U_n[d'(w) \geq 2^{(1-s)n+1}] \leq 2^{sn-n-1}$. □

One can extend the definition of HILL-type pseudo min-entropy and metric-type pseudo min-entropy by allowing $S(n)$ to be $n^{O(1)}$ and $\epsilon(n)$ to be $n^{-O(1)}$ and let us denote them by $H_\infty^{HILL}(D)$ and $H_\infty^{metric}(D)$ respectively. We can extend the result stated in Theorem 4.2 as follows.

Corollary 4.2.1. *For any ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where D_n is a distribution over Σ^n,and $s \in [0,1]$, if $H_\infty^{HILL}(D_n) \geq k$, where $k = k(n) = sn$, then $\dim(D) \geq s$.*

By Theorem 4.1 and the discussion before that, it immediately follows that $H_\infty^{HILL}(D) = H_\infty^{metric}(D)$. Thus if we consider metric-type pseudo min-entropy instead of HILL-type pseudo min-entropy, we get exactly same result as of Corollary 4.2.1.

The converse direction of the statement of Theorem 4.2 is also true if the distribution under consideration is pseudorandom. If the converse is true then we can apply any randomness extractor to get pseudorandom distribution from any distribution having high dimension [BSW03]. However, we should always be careful about the circuit size with respect to which we call the output distribution pseudorandom. Unfortunately, in general the converse is not true.

Counterexample for the converse: Suppose *one-way functions* (see Definition 2.2.1 of [Gol01]) exist, then it is well-known that we can construct a *pseudorandom generator* (see Definition 3.3.1 of [Gol01]) $G = \{G_l\}_{l\in\mathbb{N}}$ where $G_l : \Sigma^l \to \Sigma^m$ such that m is any polynomial in l, say $m = l^3$. For the construction of pseudorandom generator with polynomial stretch from any one-way function, interested reader may refer to [HILL99, VZ12]. Now consider the ensemble of distributions $D := \{(G(U_l), U_l)\}_{l\in\mathbb{N}}$. For large enough l, using the argument similar to the proof of Theorem 3.3, it can easily be shown that the distribution D has dimension 1 as the distribution on the first m bits are pseudorandom, but pseudo min-entropy is not larger than l.

4.3 Equivalence between dimension and next-bit pseudo Shannon entropy

Later, Vadhan and Zheng [VZ12] provided an alternative characterization of conditional pseudo Shannon entropy by showing an equivalence between it and *KL-hardness* (defined below). We use this alternative characterization extensively for our purpose.

Definition 4.5 (KL-hardness)**.** *Suppose (X_n, Y_n) is a $\Sigma^n \times \Sigma$-valued random variable and π be any predictor. Then π is said to be a δ-*KL-predictor *of Y_n given X_n if $\mathbf{KL}(X_n, Y_n \| X_n, C_\pi) \leq \delta$ where $C_\pi(y|x) = \pi(x, y)$ for all $x \in \Sigma^n$ and $y \in \Sigma$.*

*Moreover, for any $S = S(n) > n$ and $\delta = \delta(n) > 0$, an ensemble $Y = \{Y_n\}_{n\in\mathbb{N}}$, where each Y_n is a random variable taking value in Σ, is said to be (S, δ)-*KL-hard *given another ensemble of random variables $X = \{X_n\}_{n\in\mathbb{N}}$, where each X_n takes value in Σ^n, if for all large enough n there is no predictor π of size at most $O(S(n))$ that is a $\delta(n)$-KL-predictor of Y_n given X_n.*

The following theorem provides the equivalence among KL-hardness and conditional pseudo Shannon entropy of a distribution.

Theorem 4.3 ([VZ12])**.** *For an ensemble $(X, Y) = \{(X_n, Y_n)\}_{n\in\mathbb{N}}$ where each (X_n, Y_n) is a $\Sigma^n \times \Sigma$-valued random variable and for any $\delta = \delta(n) > 0$, $\epsilon = \epsilon(n) > 0$,*

1. *If Y is (S, δ)-KL-hard given X, then for all large enough n, Y_n has $(S'(n), \epsilon(n))$-conditional pseudo Shannon entropy at least $H(Y_n|X_n) + \delta(n) - \epsilon(n)$, where $S'(n) = S(n)^{\Omega(1)}/poly(n, 1/\epsilon(n))$.*

2. *Conversely, if for all large enough n, Y_n has $(S(n), \epsilon(n))$-conditional pseudo Shannon entropy at least $H(Y_n|X_n) + \delta(n)$, then for every $\sigma = \sigma(n) > 0$, Y is (S', δ')-KL-hard given X, where $S'(n) = \min\{S(n)^{\Omega(1)}/poly(n, \log(1/\sigma(n))), \Omega(\sigma(n)/\epsilon(n))\}$ and $\delta'(n) = \delta(n) - \sigma(n)$.*

Now we are ready to state the main theorem of this subsection which conveys the fact that the distributions with high dimensions also have high next-bit pseudo Shannon entropy.

Theorem 4.4. *For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where D_n is a distribution over Σ^n, for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$ such that $\epsilon(n) \to 0$ as $n \to \infty$, if $\dim_{S,\epsilon}(D) > 2s$, then $H^{next,S',\epsilon}(D) > k$, where $k = k(n) = sn$ and $S'(n) = S(n)^{\Omega(1)}/poly(n, 1/\epsilon(n))$.*[1]

Proof. For the sake of contradiction, let us assume that $H^{next,S',\epsilon}(D) \leq k$. Thus there exists a subset $S \subseteq \mathbb{N}$ of cardinality equals to the cardinality of $\mathbb{N}$ such that for all $n \in S$, Item 1 and Item 2 of Definition 4.4 do not hold simultaneously. On the other hand $\dim_{S,\epsilon}(D) > 2s$ and thus by following the proof of Theorem 3.1, we can say that there exists a $c \in (0,1)$ such that $\text{unpred}_{S^c,\epsilon}(D) = t > 2s$. Now consider some large enough n belongs to S and break D_n into 1-bit blocks, i.e., $D_n = (D_n^1, D_n^2, \cdots, D_n^n)$.

For any predictor $\pi : \Sigma^* \times \Sigma \to [0,1]$, let us define $\pi_i : \Sigma^{i-1} \times \Sigma \to [0,1]$ such that $\pi(x,a) = \pi_i(x,a), \forall_{x\in\Sigma^{i-1}, a\in\Sigma}$ for $1 \leq i \leq n$. Then,

$$\begin{aligned}
&\sum_{i=1}^{n} \mathbf{KL}((D_n^1, \cdots, D_n^{i-1}), D_n^i \| (D_n^1, \cdots, D_n^{i-1}), \pi_i) \\
&= \sum_{i=1}^{n} \Big[\sum_{w_i \in \Sigma^i} -\log(\pi_i(w_i[1 \cdots i-1], w_i[i])) Pr[w_i] - H(D_n^i | D_n^1 \cdots D_n^{i-1}) \Big] \\
&= \sum_{i=1}^{n} \Big[\sum_{w_i \in \Sigma^i} \mathit{loss}(\pi(w_i[1 \cdots i-1], w_i[i])) Pr[w_i] - H(D_n^i | D_n^1 \cdots D_n^{i-1}) \Big] \\
&= \sum_{w \in \Sigma^n} \text{Loss}(\pi, w) D_n(w) - H(D_n)
\end{aligned}$$

where the last equality follows from the chain rule of Shannon entropy (Proposition 2.5.1).

Now the definitions of next-bit pseudo Shannon entropy, conditional pseudo Shannon entropy and KL-predictor along with Item 1 of Theorem 4.3 imply that for any $n \in S$, there exists a predictor π of size at most $O((S(n))^c)$ such that

$$\sum_{w \in \Sigma^n} \text{Loss}(\pi, w) D_n(w) \leq (s + \epsilon(n))n.$$

Hence for any $\epsilon_1 > 0$,

$$\begin{aligned}
D_n[\text{LossRate}(\pi, w) \geq (t - \epsilon_1)] &= D_n[\text{Loss}(\pi, w) \geq (t - \epsilon_1)n] \\
&\leq \frac{\sum_{w \in \Sigma^n} \text{Loss}(\pi, w) D_n(w)}{(t - \epsilon_1)n} \quad \text{by Markov inequality} \\
&\leq \frac{s + \epsilon(n)}{t - \epsilon_1}.
\end{aligned}$$

Thus,

$$D_n[\text{LossRate}(\pi, w) \leq (t - \epsilon_1)] \geq 1 - \frac{s + \epsilon(n)}{t - \epsilon_1}.$$

As $\text{unpred}_{S^c,\epsilon}(D) = t$, by the definition, it implies that for all but finitely many n belong

[1]Note that $\Omega(1)$ constant in the expression of $S'(n)$ here is related to that appeared in Item 1 of Theorem 4.3, but not exactly the same.

to the set S,

$$1 - \frac{s+\epsilon(n)}{t-\epsilon_1} \leq \frac{1}{2} + \epsilon(n).$$

For all large enough n, the above inequality gives us $t \leq 2s + \epsilon_2$, for any $\epsilon_2 > \epsilon_1$ because $\epsilon(n) \to 0$ as $n \to \infty$. Hence $\text{unpred}_{S^c,\epsilon}(D) \leq 2s$ which is a contradiction and this completes the proof. □

The following weak converse can easily be proven.

Theorem 4.5. *For an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where D_n is a distribution over Σ^n, for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, if $H^{next,S,\epsilon}(D) > sn$, then for any $\epsilon' > 0$ $\dim_{S',\epsilon}(D) > s - \frac{1}{2} - \epsilon'$, where $S'(n) = \min\{(S(n))^{\Omega(1)}/poly(n), 1/\epsilon(n)^{\Omega(1)}\}$.*[2]

Proof. Suppose an ensemble of distributions D is given such that $H^{next,S,\epsilon}(D) > sn$. Now take any large enough n and break D_n into 1-bit blocks, i.e., $D_n = (D_n^1, D_n^2, \cdots, D_n^n)$. Let us denote $\dim_{S',\epsilon}(D) = t$ and thus by Theorem 3.1, there exists a constant $c > 1$ such that $\text{unpred}_{S'^c,\epsilon}(D) \leq t$. This implies that for infinitely many n, there exists a predictor π of size at most $O(S'^c(n))$ such that for any $\epsilon_1 > 0$,

$$D_n[\text{LossRate}(\pi, w) \leq (t+\epsilon_1)] > \frac{1}{2} + \epsilon(n).$$

Thus we can write the following,

$$\sum_{w\in\Sigma^n} \text{Loss}(\pi, w) D_n(w) \leq (t+\epsilon_1)n + (\frac{1}{2} - \epsilon(n))n.$$

We have already seen that for any predictor $\pi : \Sigma^* \times \Sigma \to [0,1]$,

$$\sum_{i=1}^{n} \mathbf{KL}((D_n^1, \cdots, D_n^{i-1}), D_n^i \| (D_n^1, \cdots, D_n^{i-1}), \pi_i) = \sum_{w\in\Sigma^n} \text{Loss}(\pi, w) D_n(w) - H(D_n).$$

Now the definitions of next-bit pseudo Shannon entropy, conditional pseudo Shannon entropy and KL-predictor along with Item 2 of Theorem 4.3 imply that for all large enough n, for every $\sigma(n) > 0$,

$$\sum_{w\in\Sigma^n} \text{Loss}(\pi, w) D_n(w) > (s - \sigma(n))n.$$

Hence,

$$\begin{aligned} &(s-\sigma(n))n < (t+\epsilon_1)n + (\frac{1}{2} - \epsilon(n))n \\ &\Rightarrow s < t + \frac{1}{2} + \epsilon_2 \end{aligned}$$

for any $\epsilon_2 > \epsilon_1$, for infinitely many large enough n and now setting $S'(n)$ to be such that $S'(n)^c$ equals to the value of $S'(n)$ mentioned in Item 2 of Theorem 4.3 concludes the proof. □

[2]Note that $\Omega(1)$ constant of the term $S(n)^{\Omega(1)}$ here is related to that appeared in Item 2 of Theorem 4.3, but not exactly the same.

It is natural to allow $S(n)$ to be $n^{O(1)}$ and $\epsilon(n)$ to be $n^{-O(1)}$ and then consider the definitions of conditional pseudo Shannon entropy, next-bit pseudo Shannon entropy (denoted as $H^{next}(X)$) and KL-hardness. For details we refer the reader to [VZ12]. Now we use the equivalence between conditional pseudo Shannon entropy and KL-hardness from [VZ12] to derive the following corollary of Theorem 4.4.

Corollary 4.3.1. *For any ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where D_n is a distribution over Σ^n, and for any $s \in [0,1]$, if $\dim(D_n) > 2s$, then $H^{next}(D) > k$ for $k = k(n) = sn$.*

In a similar fashion, we get the following as a corollary of Theorem 4.5.

Corollary 4.3.2. *For any $\epsilon > 0$, $s \in [0,1]$ and $k = k(n) = sn$, for any ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where D_n is a distribution over Σ^n, if $H^{next}(D) > k$, then $\dim(D) > s - \frac{1}{2} - \epsilon$.*

Chapter 5

Pseudorandom Extractor and Derandomization

5.1 Pseudorandom Extractors and Lower Bound

We now introduce the notion of *pseudorandom extractor* similar to the notion of randomness extractor. Intuitively, a *randomness extractor* is a function that outputs almost random (statistically close to uniform) bits from weakly random sources, which need not be close to the uniformly random source. Two distributions X and Y on a set Λ are said to be $\epsilon - close$ (statistically close) if $\max_{S \subseteq \Lambda}\{|Pr[X \in S] - Pr[Y \in S]|\} \leq \epsilon$ or equivalently $\frac{1}{2} \sum\limits_{x \in \Lambda} |Pr[X = x] - Pr[Y = x]| \leq \epsilon$.

Definition 5.1 (Deterministic Randomness Extractor)**.** *For any $\epsilon = \epsilon(n) > 0$, a family of functions $E = \{E_n\}_{n \in \mathbb{N}}$ where $E_n : \Sigma^n \to \Sigma^{m(n)}$ is said to be a deterministic ϵ-extractor for a class of ensemble of distributions $\mathcal{C}$ if for every ensemble of distributions $X = \{X_n\}_{n \in \mathbb{N}}$ in $\mathcal{C}$, where X_n is on Σ^n, for all large enough n, the distribution $E_n(X_n)$ is $\epsilon(n)$-close to $U_{m(n)}$.*

Likewise, a seeded ϵ-extractor is defined and the only difference is that now it takes a $d(n)$-bit string chosen according to $U_{d(n)}$, as an extra input. Before going further, we mention that for ease of presentation, now onwards we will only talk about the definitions and results derived so far related to pseudorandomness and dimension by considering all polynomial size circuits and inverse polynomial bias. We now define the notion of a pseudorandom extractor, the purpose of which is to extract out pseudorandom distribution from a given distribution.

Definition 5.2 (Pseudorandom Extractor)**.** *A family of functions $E = \{E_n\}_{n \in \mathbb{N}}$ where $E_n : \Sigma^n \to \Sigma^{m(n)}$ is said to be a* deterministic pseudorandom extractor *for a class of ensemble of distributions $\mathcal{C}$ if for every ensemble of distributions $X = \{X_n\}_{n \in \mathbb{N}}$ in $\mathcal{C}$, where X_n is on Σ^n, $E(X)$ is pseudorandom.*

A family of functions $E = \{E_n\}_{n \in \mathbb{N}}$ where $E_n : \Sigma^n \times \Sigma^{d(n)} \to \Sigma^{m(n)}$ is said to be a seeded pseudorandom extractor *for a class of ensemble of distributions $\mathcal{C}$ if for every ensemble of distributions $X = \{X_n\}_{n \in \mathbb{N}}$ in $\mathcal{C}$, $E(X, U_d)$ is pseudorandom.*

In this section, we will concentrate on the class of distributions having dimension at least s. It is clear from the results stated in Section 4.2 that this class of distribution is a strict superset of the class of distributions with HILL-type pseudo min-entropy at least sn, for which any randomness extractor will act as a pseudorandom extractor [BSW03]. Thus it is natural to ask the following.

Question 5.1. *For any $s \in (0, 1]$, does there exist a deterministic/seeded pseudorandom extractor for the class of ensemble of distributions having dimension at least s?*

Just like the the case of randomness extraction, one can easily argue that deterministic pseudorandom extraction is not possible[1]. The next natural question is what the lower bound on the seed length will be. We answer this question in the following theorem.

Theorem 5.1. *Suppose for any $s \in (0, 1]$, $E = \{E_n\}_{n\in\mathbb{N}}$ where $E_n : \Sigma^n \times \Sigma^{d(n)} \to \Sigma^{m(n)}$ be a seeded pseudorandom extractor for the class of ensemble of distributions having dimension at least s and for some $\delta > 0$, $m(n) = (sn)^\delta$. Then $d(n) = \Omega(\log n)$.*

Proof. For the sake of contradiction, let us assume that $d(n) = o(\log n)$. Now by doing a walk according to the output distribution on an odd-length cycle, we achieve the following claim.

Claim 5.1.1. *There is a deterministic $\frac{1}{\sqrt[4]{m}}$-extractor $E' = \{E'_m\}_{m\in\mathbb{N}}$ where $E'_m : \Sigma^m \to \Sigma^{\frac{\log m}{4}}$ for all pseudorandom ensemble of distributions.*

This claim follows from the stronger result stated in Theorem 5.3. Now construct the following function $Ext_n : \Sigma^n \times \Sigma^{d(n)} \to \Sigma^{c\log n}$ for some constant $c > 0$ such that $Ext_n(x, y) = E'_n(E_n(x, y))$ for all $x \in \Sigma^n, y \in \Sigma^{d(n)}$. The function Ext is a seeded $\frac{1}{(sn)^{\delta/4}}$-extractor with $d(n) = o(\log n)$, but it is well known due to [RTS00](Theorem 1.9) that any such randomness extractor must satisfy $d(n) = \Omega(\log n)$ and hence we get a contradiction. □

However, the question on constructing an *explicit* or polynomial time computable seeded pseudorandom extractor with seed length $O(\log n)$ is still open and next, we formally pose this question.

Question 5.2. *For any $s \in (0, 1]$, can one construct a seeded pseudorandom extractor $E = \{E_n\}_{n\in\mathbb{N}}$ where $E_n : \Sigma^n \times \Sigma^{d(n)} \to \Sigma^{m(n)}$ in polynomial time, for the class of ensemble of distributions having dimension at least s such that $m(n) = (sn)^\delta$ for some $\delta > 0$ and $d(n) = O(\log n)$?*

Note that it is important to consider dimension in the statement of Question 5.1 and 5.2, because if we consider strong dimension instead of dimension then sometimes it might be just impossible to extract out pseudorandom distributions. For example one can consider the ensemble of distributions mentioned in the proof of Theorem 3.7 where however strong dimension is 1, as for infinitely many n, support of D_n contains just a single string, thus one cannot hope to get any pseudorandom distribution out of it. In the next part of this section, we will see a special type of nonpseudorandom source and give an explicit construction of deterministic pseudorandom extractor for that particular type of source. Before proceeding further, we want to mention that it is also very interesting to consider *nonpseudorandom distributions samplable by poly-size circuits* and we will discuss on the existence of extractor for that particular source in Section 5.1.2.

[1]Suppose $E = \{E_n\}_{n\in\mathbb{N}}$ with $E_n : \Sigma^n \to \Sigma$ is a deterministic pseudorandom extractor, then for all n, there exists $x \in \Sigma$ such that $|E_n^{-1}(x)| \geq 2^{n-1}$. Thus E is not a pseudorandom extractor for a source D that is a uniform distribution on $E_n^{-1}(x)$ for all n and by Lemma 4.2.1, $\dim(D) \geq s$ for any $s < 1$.

5.1.1 Deterministic Pseudorandom Extractor for Nonpseudorandom Bit-fixing Sources

In Section 3.3 while proving Theorem 3.3, we have introduced a special type of nonpseudorandom distribution which looks similar to the (n, k)-*bit-fixing source* defined as a distribution X over Σ^n such that there exists a subset $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ where all the bits at the indices of I are independent and uniformly chosen and rest of the bits are completely fixed. This distribution was introduced by Chor *et al.*[CGH$^+$85]. Now we define an analogous notion for the class of nonpseudorandom distributions, which we term *nonpseudorandom bit-fixing sources.*

Definition 5.3 (Nonpseudorandom Bit-fixing Source)**.** *Let $s \in (0, 1)$. An ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$, where D_n is an distribution over Σ^n, with dimension s is an (n, s)-*nonpseudorandom bit-fixing source *if for all n there exists a subset $I = \{i_1, \ldots, i_{\lceil sn\rceil}\} \subseteq \{1, \ldots, n\}$ such that all the bits at the indices of I come from a pseudorandom distribution and rest of the bits are fixed.*

We devote the rest of the section to achieve an affirmative answer to the question of constructing deterministic pseudorandom extractor for the nonpseudorandom bit-fixing sources. For this purpose, we show that a careful analysis of the technique used in the construction of the deterministic randomness extractor for bit-fixing random sources by Gabizon, Raz and Shaltiel [GRS04] will lead us to the desired deterministic pseudorandom extractor.

Theorem 5.2. *For any $s \in (0, 1]$, there is an explicit deterministic pseudorandom extractor $E = \{E_n : \Sigma^n \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$, for all (n, s)-nonpseudorandom bit-fixing sources having polynomial-size support where $m(n) = (sn)^{\Omega(1)}$.*

We first extract $O(\log sn)$ amount of almost random bits and then use the same as seed in the seeded extractor. To use the seeded extractor, we modify the source such that it becomes independent of the random bits extracted. Before going into the exact details of the proof, we first discuss the ingredients required in the proof of the theorem.

Pseudorandom walk and extracting a few random bits

Kamp and Zuckerman [KZ03] use a technique of random walk on odd-length cycles to extract almost random bits from a bit-fixing source. We adapt this to extract $O(\log sn)$ almost random bits from a (n, s)-nonpseudorandom bit-fixing source.

Theorem 5.3. *Let $s \in [0, 1]$, $k = \lceil sn\rceil$. Then there is a deterministic $\frac{1}{\sqrt[4]{k}}$-extractor $E = \{E_n : \Sigma^n \to \Sigma^{\frac{\log k}{4}}\}_{n\in\mathbb{N}}$ for all (n, s)-nonpseudorandom bit-fixing sources.*

We prove the above theorem using the property of *pseudorandom walk* together with the fact that the second largest eigenvalue of a l length odd cycle is $\cos(\pi/l)$. Note that a corollary of the above theorem is the claim used in the proof of Theorem 5.1. Before proving the above theorem, we state two lemmas required for the proof. The first is a very special case of a lemma given in [KZ03] which was restated in [GRS04].

Lemma 5.1.1 ([GRS04])**.** *Let $n \in \mathbb{N}$, $k \leq n$ and $s \in [0, 1]$. Let G be an odd length cycle having M vertices and having second largest eigenvalue λ. If we take a walk on G*

according to the bits from a (n,k)-bit-fixing source, starting from any fixed vertex, then at the end of the n step of the walk, the distribution P on the vertices will be $\frac{1}{2}\lambda^k\sqrt{M}$-close to U_M.

Now we prove a similar result for (n,s)-nonpseudorandom bit-fixing source using the property of *pseudorandom walk*. The idea of pseudorandom walk was also used previously in the domain of space bounded computation by Reingold *et al.* [RTV06].

Lemma 5.1.2. *Let $s \in [0,1]$ and $k = \lceil sn \rceil$. Suppose G is an odd length cycle having M vertices and having second largest eigenvalue λ. If we take a walk on G according to the bits from a (n,s)-nonpseudorandom bit-fixing source starting from any fixed vertex, then for all large enough n, at the end of the n step of the walk, the distribution Q on the vertices will be $\frac{1}{2}(\lambda^k + \sqrt{M}\epsilon(n))\sqrt{M}$-close to U_M, where M is polynomial in n and $\epsilon(n) < 1/n^c$ for any constant $c > 0$.*

Proof. Let π be the stationary distribution on the vertices and since we consider an odd length cycle (a 2-regular graph), the stationary distribution is the uniform distribution on M vertices. Suppose we take a n step walk on the graph G starting from any vertex v according to the bits from a (n,k)-bit-fixing source, where $k = \lceil sn \rceil$ and the probability vector on the vertices at the end of the walk is $P = \begin{pmatrix} p_1 & p_2 & \dots & p_M \end{pmatrix}$. Now we take a n step walk on the graph G starting from the same vertex v according to the bits from a (n,s)-nonpseudorandom bit-fixing source and the probability vector on the vertices at the end of the walk is $Q = \begin{pmatrix} q_1 & q_2 & \dots & q_M \end{pmatrix}$, where $\forall_{1\le i\le M}$, $q_i \le p_i + \epsilon(n)$ and $\epsilon(n) < 1/n^c$ for any constant $c > 0$. This bound on q_i can be justified as follows.

Note that the only difference between (n,s)-nonpseudorandom bit-fixing source and (n,k)-bit-fixing source is that on the set I, in (n,k)-bit-fixing source, we have U_k instead of pseudorandom distribution (say D) on Σ^k. Also observe that actually P and Q are the distributions on vertices at the end of a k step walk, where the walk was started from the vertex v and done according to the bits coming from U_k and D respectively, because a step according to a fixed bit will not change the output distribution and in a (n,k)-bit-fixing source (also in a (n,s)-nonpseudorandom bit-fixing source), all the $n-k$ bits are fixed. For a step according to a fixed bit gives rise to a transition matrix that is actually a permutation matrix and thus it leaves the distance from uniform unchanged. Hence, if the bound on $q_i, \forall_{1\le i\le M}$ is not true then we can use this k step walk on G as a polynomial (polynomial in k) time algorithm to distinguish between U_k and D. Thus,

$$\begin{aligned} ||q-\pi||^2 = \sum_{i=1}^{M}(q_i - \frac{1}{M})^2 &\le \sum_{i=1}^{M}(p_i + \epsilon(n) - \frac{1}{M})^2 \\ &= ||p-\pi||^2 + M\epsilon(n)^2 \\ &\le \lambda^{2k} + M\epsilon(n)^2 \\ &\le (\lambda^k + \sqrt{M}\epsilon(n))^2. \end{aligned}$$

□

The above lemma together with the fact that the second largest eigenvalue of a l length odd cycle is $\cos(\pi/l)$, implies Theorem 5.3.

Proof of Theorem 5.3. Let us take an odd cycle G with $M = \sqrt[4]{k}$ vertices. The second largest eigenvalue of G is $\cos(\frac{\pi}{\sqrt[4]{k}})$. Now take a walk starting from a fixed vertex of G according to the bits from (n, s)-nonpseudorandom bit-fixing source and finally output the vertex number of the graph G. Thus we get $\frac{\log k}{4}$ bits. From Lemma 5.1.2, we reach distance $\frac{1}{2}((\cos(\frac{\pi}{\sqrt[4]{k}}))^k + \sqrt[8]{k}\epsilon(n))\sqrt[8]{k}$ from uniform.

By the Taylor expansion of the cosine function, for $0 < x < 1$, $\cos(x) < 1 - \frac{x^2}{2} + \frac{x^4}{24}$. Therefore, $\left(\cos\left(\frac{\pi}{\sqrt[4]{k}}\right)\right)^k < (1 - \frac{\pi^2}{4\sqrt{k}})^k < (\exp^{-\frac{\pi^2}{4}})^{\sqrt{k}} < 4^{-\sqrt{k}}$. Hence, $\frac{1}{2}((\cos(\frac{\pi}{\sqrt[4]{k}}))^k + \sqrt[8]{k}\epsilon(n))\sqrt[8]{k} < \frac{1}{\sqrt[4]{k}}$. Thus we get distribution of $\frac{\log k}{4}$ bit strings which is $\frac{1}{\sqrt[4]{k}}$-close to uniform in statistical distance. □

Sampling and Partitioning with a short seed

Here we restate some of the results on sampling and partitioning used in construction of deterministic extractor for bit-fixing sources from [GRS04]. Let $S \subseteq [n]$ be some subset of size k. Now we consider a process of generating a subset $T \subseteq [n]$ such that $k_{min} \leq |S \cap T| \leq k_{max}$ and this process is known as *Sampling*.

Definition 5.4. *A function Samp* : $\Sigma^t \to P([n])$ *is called a* $(n, k, k_{min}, k_{max}, \delta)$*-sampler if for any subset* $S \subseteq [n]$*, where* $|S| = k$*,* $Pr_{w \in_R U_t}[k_{min} \leq |Samp(w) \cap S| \leq k_{max}] \geq 1 - \delta$

Now consider a similar process known as *Partitioning*, the task of which is to partition $[n]$ into m distinct subsets $T_1, T_2, \cdots, T_m$ such that for every $1 \leq i \leq m$, $k_{min} \leq |S \cap T_i| \leq k_{max}$.
According to [GRS04], the above two processes can be performed using only *a few* random bits.

Lemma 5.1.3 ([GRS04]). *For any constant* $0 < \alpha < 1$*, there exist constants* $c > 0, 0 < b < 1$ *and* $\frac{1}{2} < e < 1$ *such that for any* $n \geq 16$ *and* $k \geq (\log n)^c$*, there is an explicit construction of a function Samp* : $\Sigma^t \to P([n])$ *which is a* $(n, k, \frac{k^e}{2}, 3k^e, O(k^{-b}))$*-sampler, where* $t = \alpha \log k$*.*

Lemma 5.1.4 ([GRS04]). *For any constant* $0 < \alpha < 1$*, there exist constants* $c > 0, 0 < b < 1$ *and* $\frac{1}{2} < e < 1$ *such that for any* $n \geq 16$ *and* $k \geq (\log n)^c$*, there is an explicit construction that uses only* $\alpha \log k$ *random bits and partition* $[n]$ *into* $m = O(k^b)$ *many subsets* $T_1, T_2, \cdots, T_m$ *such that for any subset* $S \subseteq [n]$*, where* $|S| = k$*,* $Pr[\forall 1 \leq i \leq m, \frac{k^e}{2} \leq |T_i \cap S| \leq 3k^e] \geq 1 - O(k^{-b})$*.*

Generating an independent seed

In this subsection, we see the way of obtaining a short seed from a nonpseudorandom bit-fixing source so that we can use them in a seeded pseudorandom extractor to extract out almost all the pseudorandom part from the source. The main problem of using this short seed in a seeded pseudorandom extractor is that the already obtained seed is dependent on the main distribution. Now we describe that this problem can be removed in the case of nonpseudorandom bit-fixing sources. Even though the result is analogous to [GRS04], the proofs differ in essential details.

Definition 5.5 (Seed Obtainer). *A family of functions $F = \{F_n : \Sigma^n \to \Sigma^n \times \Sigma^{d(n)}\}_{n\in\mathbb{N}}$ is said to be a (s, s', ρ)-seed obtainer ($s' \leq s$) if for every (n, s)-nonpseudorandom bit-fixing source $X = \{X_n\}_{n\in\mathbb{N}}$, the distribution $R = \{R_n = F_n(X_n)\}_{n\in\mathbb{N}}$ can be written as $R = \eta Q + \sum_a \alpha_a R_a$[2] for $\eta = \eta(n) > 0$, $\alpha_a = \alpha_a(n) > 0$ and $\eta(n) + \sum_a \alpha_a(n) = 1$ such that $\eta(n) \leq \rho(n)$ and for every a, there exists a (n, s')-nonpseudorandom bit-fixing source Z_a such that for all large enough n, $\{R_a\}_n$ is $\rho(n)$-close to $\{Z_a\}_n \otimes U_{d(n)}$.*

In the above definition, by $\{Z_a\}_n \otimes U_{d(n)}$, we mean the product of two distributions $\{Z_a\}_n$ and $U_{d(n)}$. From the above definition it is clear that given a seed obtainer and a seeded pseudorandom extractor for nonpseudorandom bit-fixing sources, we can easily construct a deterministic pseudorandom extractor. The following theorem provides us the details of such construction, where the correctness follows from the properties of our proposed notion of dimension described in Section 3.3.

Theorem 5.4. *Suppose $F = \{F_n : \Sigma^n \to \Sigma^n \times \Sigma^{d(n)}\}_{n\in\mathbb{N}}$ is a (s, s', ρ)-seed obtainer, where $\rho(n) \leq \frac{1}{(sn)^c}$ for some constant $c > 0$ and $E' = \{E'_n : \Sigma^n \times \Sigma^{d(n)} \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$ is a seeded pseudorandom extractor for (n, s')-nonpseudorandom bit-fixing sources, where $m(n) = (sn)^{\Omega(1)}$. Then the function $E = \{E_n : \Sigma^n \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$ defined as $E_n(x) = E'_n(F_n(x))$ for all $x \in \Sigma^n$, is a deterministic pseudorandom extractor for (n, s)-nonpseudorandom bit-fixing sources.*

Proof. By the definition of the seed obtainer, we can write $E(X) = \eta E'(Q) + \sum_a \alpha_a E'(R_a) = \eta E'(Q) + (1-\eta)Y$, for some ensemble of distributions Y. Now by Lemma 3.3.2, for all a, $E'(R_a)$ is pseudorandom and as a consequence, by Lemma 3.3.3, Y is pseudorandom as well. Then using Lemma 3.3.2, we can argue that $E(X)$ is also pseudorandom as $\eta \leq \frac{1}{(sn)^c}$, for some constant $c > 0$. □

Now we give an explicit construction of (s, s', ρ)-seed obtainer, which is crucial in the later part of this paper. To understand the notion of *sampler* used in the following theorem, the readers may refer to the last subsection.

Theorem 5.5. *Let $Samp = \{Samp_n : \Sigma^{t(n)} \to P([n])\}_{n\in\mathbb{N}}$ where $Samp_n$ be a $(n, \lceil sn\rceil, \lceil s_1 n\rceil, \lceil s_2 n\rceil, \delta(n))$-sampler and $E = \{E_n : \Sigma^n \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$ with $m(n) > t(n)$ be a deterministic ϵ-extractor for (n, s_1)-nonpseudorandom bit-fixing sources, where $\epsilon(n) < 1/n^c$ for any constant $c > 0$. Then there is an explicit (s, s', ρ)-seed obtainer $F = \{F_n : \Sigma^n \to \Sigma^n \times \Sigma^{d(n)}\}_{n\in\mathbb{N}}$, where $d(n) = m(n) - t(n)$, $s' = s - s_2$, and $\rho(n) = \max\{\epsilon(n) + \delta(n), \sqrt{\epsilon(n)}2^{t(n)+1}\}$.*

The construction of the seed obtainer is the same as that mentioned in [GRS04], however the proof requires a slightly different argument.

Proof. The construction of F mentioned in the theorem is as follows:

1. Given $x \in \Sigma^n$, compute $E_n(x)$. Denote the first $t(n)$ bits of $E_n(x)$ by $E^1_n(x)$ and the last $(m(n) - t(n))$ bits by $E^2_n(x)$;

2. Compute $Samp_n(E^1_n(X))$ and denote it as T;

[2]It means for all n, $R_n = \eta(n)Q_n + \sum_a \alpha_a(n)\{R_a\}_n$

3. Let $x' = x_{[n]\setminus T}$ and $y = E_n^2(x)$. If $|x'| < n$, pad it with zeros to get n-bit long string. Now output x', y.

Note that the above construction is the same as the construction of seed obtainer given in [GRS04]. However, the proof is not the same and more specifically the proof of the next claim differs from that of the similar claim made in [GRS04]. Here, in the proof we use the properties of pseudorandomness and the fact that the distribution under consideration has polynomial-size support.

Let X be a (n, s)-nonpseudorandom bit-fixing source and now consider any large enough n and let I be the set of indices at which the bits are not fixed. For a string $a \in \Sigma^{t(n)}$, T_a denotes $Samp_n(a)$ and T'_a denotes $[n] \setminus Samp_n(a)$. Given a string $x \in \Sigma^n$, x_a denotes x_{T_a} and x'_a denotes n-bit string obtained by padding $x_{T'_a}$ with zeros. Let $X' = X'_{E_n^1(X_n)}$ and $Y = E_n^2(X_n)$. We say that a string $a \in \Sigma^{t(n)}$ *correctly splits* X_n if $\lceil s_1 n \rceil \le |I \cap T_a| \le \lceil s_2 n \rceil$.

Claim 5.1.2. *For every $a \in \Sigma^{t(n)}$ which correctly splits X_n, $(X'_a, E_n(X_n))$ is $\epsilon(n)$-close to $(X'_a \otimes U_{m(n)})$, where $\epsilon(n) < 1/n^c$ for any constant $c > 0$.*

Proof. Let $|Samp_n(a)| = l$. Given a string $\sigma \in \Sigma^l$ and a string $\sigma' \in \Sigma^{n-l}$, we define $[\sigma; \sigma']$ as follows:
Suppose l indices of T_a are $i_1 < \cdots < i_l$ and the $(n-l)$ indices of T'_a are $i'_1 < \cdots < i'_{n-l}$. The string $[\sigma; \sigma'] \in \Sigma^n$ is defined as:

$$[\sigma; \sigma']_i = \begin{cases} \sigma_j & i \in T_a \text{ and } i_j = i \\ \sigma'_j & i \in T'_a \text{ and } i'_j = i \end{cases}$$

In this notation, we denote $X_n = [X_a; X'_a]$. Now consider the distribution $(X'_a, E_n(X_n)) = (X'_a, E_n([X_a; X'_a]))$. For every $b \in \Sigma^{n-l}$, we consider the event $\{X'_a = b\}$. As a correctly splits X_n, there are at least $\lceil s_1 n \rceil$ "good" indices in T_a. Now fix some $b \in \Sigma^{n-l}$ such that $X_n[X'_a = b] > 0$.

Now we claim that for all subsets $B \subseteq \Sigma^{n-l}$ where $\forall_{b \in B}, X_n[X'_a = b] > 0$, there exists a $b' \in B$ such that the distribution $([X_a; X'_a] | X'_a = b')$ forms an (n, s_1)-nonpseudorandom bit-fixing source if for some constant $c > 0$, $\epsilon'(n) \ge 1/n^c$ and

$$\sum_{b \in B} X_n[X'_a = b] > \epsilon'(n).$$

For the sake of contradiction, let us assume that the above claim is not true. It means that there exists a subset $J \subseteq \Sigma^{n-l}$, where

i. $\forall_{b \in J}$, $X_n[X'_a = b] > 0$,

ii. $\sum_{b \in J} X_n[X'_a = b] > \epsilon'(n)$ where $\epsilon'(n) \ge 1/n^c$, for some constant $c > 0$, and also

iii. for all $b \in J$, the distributions $([X_a; X'_a] | X'_a = b)$ are not forming (n, s_1)-nonpseudorandom bit-fixing sources.

Now let us consider only the "good" positions which are $\lceil sn \rceil$ many in X and at least $\lceil s_1 n \rceil$ many in $([X_a; X'_a] | X'_a = b)$. So the above assumption implies that the ensemble of distributions formed by considering those $\lceil s_1 n\rceil$ bits (this part of the string b is denoted as $b_{\lceil s_1 n\rceil}$) in $([X_a; X'_a]|X'_a = b)$ is not pseudorandom, i.e., it has its corresponding distinguishing circuits C_b. If this is the case, then the circuit C (by hard-wiring the good random bits) corresponding to the following algorithm A, will act as a distinguishing circuit for the pseudorandom distribution P on $\lceil sn\rceil$ many bits; which is a contradiction. The algorithm A is as follows: on input $y \in \{0,1\}^{\lceil s_1 n\rceil}$, if $y_{\lceil s_1 n\rceil} = b_{\lceil s_1 n\rceil}$ for any $b \in J$, then return $C_b(y_{\lceil s_1 n\rceil})$; otherwise return 0 or 1 uniformly. And thus clearly,

$$|P[A[y] = 1] - U_{\lceil s_1 n\rceil}[A[y] = 1]| > 1/n^c.$$

Circuit C is nothing but the combination of all the circuits C_b, for $b \in J$, each of which is of polynomial size. Now as $\forall_{b\in J},\ X_n[X'_a = b] > 0$ and by our assumption that the distribution under consideration has polynomial-size support (see statement of Theorem 5.2), the support of the subset J is at most polynomial. Hence the circuit C is of polynomial size. Note that this is the only place where we use the fact that the distribution under consideration is of polynomial-size support.

So, we can write,

$$\begin{aligned}&\frac{1}{2}\sum_{b,c} |Pr[(X'_a, E(X)) = (b,c)] - Pr_{(X'_a \otimes U_{m(n)})}[b,c]| \\ &= \frac{1}{2}\sum_{b,c} |Pr[X'_a = b]Pr[E_n(X_n) = c|X'_a = b] - Pr[X'_a = b]Pr_{U_{m(n)}}[c]| \leq \epsilon(n)\end{aligned}$$

where $\epsilon(n) < 1/n^c$ for any constant $c > 0$. The first inequality follows from the fact that we can split the sum in two parts one in which $([X_a; X'_a]|X'_a = b)$'s are not (n, s_1)-nonpseudorandom bit-fixing sources and another in which $([X_a; X'_a]|X'_a = b)$'s are at least (n, s_1)-nonpseudorandom bit-fixing sources. □

Next we mention a claim from [GRS04] that makes comment on independence of the pair $(X'_a, E^2_n(X_n))$ conditioned on the event $E^1_n(X_n) = a)$.

Claim 5.1.3 ([GRS04])**.** *For every fixed $a \in \Sigma^{t(n)}$ that correctly splits X_n, the distribution $((X'_a, E^2_n(X)) \mid E^1_n(X) = a)$ is $\epsilon(n)2^{t+1}$-close to $(X'_a \otimes U_{m(n)-t(n)})$.*

Note that as a correctly splits X_n, so X'_a forms a $(n, s-s_2)$-nonpseudorandom bit-fixing source.

The rest of the proof follows directly from the proof of correctness of the construction of seed obtainer given in [GRS04] with the following parameters $k = \lceil sn\rceil$, $k_{min} = \lceil s_1 n\rceil$, $k_{max} = \lceil s_2 n\rceil$. □

A seeded pseudorandom extractor

In this subsection, we discuss about how we can extract $(sn)^{\Omega(1)}$ many pseudorandom bits using $O(\log sn)$ random bits. In the next subsection, we will use this seeded pseudorandom extractor and the techniques discussed in the previous subsections, to construct deterministic extractor. The construction of seeded pseudorandom generator given in the proof of the following theorem is same as that of the seeded randomness extractor given in [GRS04]. However, the analysis is quite different and uses some of the properties of dimension.

Theorem 5.6. *For an $s \in (0,1)$ and any constant $0 < \alpha < 1$, there exist constants $c > 0, 0 < b < 1$ such that there is an explicit function $E = \{E_n : \Sigma^n \times \Sigma^{d(n)} \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$ which acts as a seeded pseudorandom extractor for (n, s)-nonpseudorandom bit-fixing sources with $d(n) = \alpha \log sn$ and $m(n) = \Omega((sn)^b)$.*

Proof. Let X be a (n, s)-nonpseudorandom bit-fixing source and for some large enough n, x be a string sampled by X_n. The description of the extractor $E_n(x, y)$ is as follows:

1. According to Lemma 5.1.4 provided in Section 5.1.1, using y as seed, we obtain a partition of $[n]$ into $m(n) = \Omega((sn)^b)$ many sets $T_1, T_2, \cdots, T_{m(n)}$ with the parameter α;

2. For $1 \le i \le m(n)$, compute $z[i] = \oplus_{j\in T_i} x[j]$;

3. Output $z = z[1]z[2] \cdots z[m(n)]$.

Let $I \subseteq [n]$ be the set of indices at which the bits are not fixed and let Z_n be the distribution of the output strings. We need to show that $Z = \{Z_n\}_{n\in\mathbb{N}}$ is pseudorandom.

Let A_n be the event $\{\forall i,\ |T_i \cap I| \neq 0\}$ and $A_n^c = \{\exists i,\ |T_i \cap I| = 0\}$ be the complement event. According to Lemma 5.1.4, $Pr[A_n] \ge 1 - O((sn)^{-b})$. Now we can write the output distribution as

$$Z_n = Pr[A_n](Z_n|A_n) + Pr[A_n^c](Z_n|A_n^c)$$

and hence due to Lemma 3.3.2, Z is pseudorandom. □

Deterministic pseudorandom extractor

Now it only remains to combine all the components we discussed so far to build the final deterministic pseudorandom extractor mentioned in Theorem 5.2. We first extract $O(\log sn)$ amount of almost random bits by Theorem 5.3 and then use the same as seed in the seeded extractor described in Theorem 5.6. To use the seeded extractor it is required to modify the source such that it becomes independent of the random bits extracted using Theorem 5.3. For that purpose, we use the technique developed in Section 5.1.1 and this concludes the proof of Theorem 5.2.

Proof of Theorem 5.2. Due to Lemma 5.1.3, we have a $(n, sn, \frac{(sn)^e}{2}, 3(sn)^e, (sn)^{-\Omega(1)})$-sampler $Samp_n : \Sigma^{t(n)} \to P([n])$, where $t(n) = \frac{\log sn}{32}$ and $e > \frac{1}{2}$. From Theorem 5.3, we have a deterministic $\frac{1}{\sqrt[4]{s'n}}$-extractor $E^* = \{E_n^* : \Sigma^n \to \Sigma^{m'}\}_{n\in\mathbb{N}}$ for (n, s')-nonpseudorandom bit-fixing sources where for all large enough n, $s'n \ge \frac{(sn)^e}{2}$ and $m'(n) = \frac{\log s'n}{4}$. Now we use Theorem 5.5 to get (s, s'', ρ)-seed obtainer $F = \{F_n : \Sigma^n \to \Sigma^n \times \Sigma^{m'(n)-t(n)}\}_{n\in\mathbb{N}}$ where for all large enough n, $s''n \ge 3(sn)^e$ and $\rho(n) = \frac{1}{(sn)^p}$, for some constant p. According to Theorem 5.6, we have a seeded pseudorandom extractor $E' = \{E_n' : \Sigma^n \times \Sigma^{d(n)} \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$ with $d(n) = \frac{\log sn}{32}$ and $m(n) = (sn - s''n)^{\Omega(1)}$ for $(n, s - s'')$-nonpseudorandom bit-fixing sources. Since $m'(n) = \frac{\log s'n}{4} \ge \frac{\log sn}{16} = t(n) + d(n)$, we use F and E' in Theorem 5.4 to construct a deterministic pseudorandom extractor $E = \{E_n : \Sigma^n \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$. For a large enough n, $m(n) = (sn - s''n)^{\Omega(1)} = (sn)^{\Omega(1)}$ and this completes the proof. □

Alternative Construction of Deterministic Pseudorandom Extractor for Nonpseudorandom Bit-fixing Sources

Here we provide a simple alternative proof of constructibility of deterministic pseudorandom extractor for nonpseudorandom bit-fixing sources. Let us first state the theorem.

Theorem 5.7. *For any $s \in (0,1]$, there is an explicit deterministic pseudorandom extractor $E = \{E_n : \Sigma^n \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$ for all (n,s)-nonpseudorandom bit-fixing sources, where $m(n) = (sn)^{\Omega(1)}$.*

Note that the statement of the above theorem is stronger than that of Theorem 5.2 because now we do not have any restriction on the support size of the distribution whereas in Theorem 5.2 we need the distribution to have polynomial-size support. Before going into the proof we reformulate the main result of [GRS04].

Theorem 5.8 ([GRS04])**.** *For any $s \in (0,1]$ and for every constant $0 < \delta < 1/2$ there exists an n' such that for any $n \geq n'$, there is an explicit deterministic $\frac{1}{2^{n^\delta}}$-extractor $E = \{E_n : \Sigma^n \to \Sigma^{m(n)}\}_{n\in\mathbb{N}}$ for all ensemble $(n, \lceil sn\rceil)$-bit-fixing sources, where $m(n) = (sn)^{\Omega(1)}$.*

We argue that the extractor E appeared in the statement of the above theorem also act as a pseudorandom extractor for (n,s)-nonpseudorandom bit-fixing sources.

Proof of Theorem 5.7. Suppose E mentioned in Theorem 5.8 is not a pseudorandom extractor for (n,s)-nonpseudorandom bit-fixing sources. Thus there exists a polynomial-size (polynomial in $m(n)$) circuit C such that for a (n,s)-nonpseudorandom bit-fixing source $X = \{X_n\}_{n\in\mathbb{N}}$ and for some constant $c > 0$,

$$|E_n(X_n)[C(x) = 1] - U_{m(n)}[C(x) = 1]| > \frac{1}{n^c}.$$

Now suppose underlying pseudorandom distribution in $X = \{X_n\}_{n\in\mathbb{N}}$ is $Y = \{Y_n\}_{n\in\mathbb{N}}$. Then using the extractor function E_n and the circuit C we can build another polynomial-size (polynomial in n and thus polynomial in sn as well because s is a constant) circuit C' by specifying the index set I where the pseudorandom distribution Y_n lies such that the following holds,

$$|Y_n[C'(x) = 1] - U_{\lceil sn\rceil}[C'(x) = 1]| > \frac{1}{n^c}$$

which contradicts the fact that $Y = \{Y_n\}_{n\in\mathbb{N}}$ is pseudorandom. This completes the proof. □

5.1.2 Discussion on Pseudorandom Extractor for Nonpseudorandom Samplable Distributions

Another interesting special kind of source is samplable distributions studied by Trevisan and Vadhan [TV00]. In a natural way, one can extend the definition of samplable distribution to nonpseudorandom distribution as follows: for any $s \in (0,1]$, an ensemble of distributions $D = \{D_n\}_{n\in\mathbb{N}}$ is said to be s-nonpseudorandom samplable by circuit of size $S = S(n) > n$ if for all large enough n, there exists a circuit C of size at most $S(n)$ that samples from D_n and $\dim(D) = s$. Observe that the negative results for deterministic randomness extractor in case of samplable distributions will also applicable for deterministic pseudorandom extractor in case of s-nonpseudorandom samplable distribution. By Lemma 4.2.1, if $H_\infty(D_n) \geq n - 1$ for all large enough n, then $\dim(D) \geq s$ for any $s < 1$. Now by applying the argument in [TV00], we get the following.

Theorem 5.9. *Suppose $E = \{E_n : \Sigma^n \to \Sigma\}_{n\in\mathbb{N}}$ is a family of functions computable in time $T(n)$ such that E is a deterministic pseudorandom extractor for ensemble of distributions that are s-nonpseudorandom samplable by circuit of size $S(n)$ for any $s < 1$. Then there is a language in $DTIME(T(n))$ of circuit complexity at least $\Omega(S(n))$.*

The existence of deterministic pseudorandom extractors implies separations between deterministic complexity classes and non-uniform circuit classes that are not yet known. So one might have to consider some complexity theoretic assumptions like in [TV00] to construct deterministic pseudorandom extractor. However, we do not think construction with such strong assumption like in [TV00] will be interesting in this case as it is known that certain hardness assumption already leads to a construction of optimal pseudorandom generator (See Section 5.2). Nevertheless, it is natural to ask the question of constructing explicit extractor using $O(\log n)$ amount of extra randomness. We do not know any such result so far, but in the next section we will see that if some distribution is samplable using very few ($O(\log n)$) random bits, then it is possible to extract out all the pseudorandom bits using extra $O(\log n)$ random bits.

5.2 Approaching Towards P = BPP

We now show that if there is an exponential time computable algorithm $G = \{G_n\}_{n\in\mathbb{N}}$ with $G_n : \Sigma^{O(\log n)} \to \Sigma^n$ where the output distribution has dimension s ($s > 0$), then this will imply P=BPP. We refer to this algorithm G as *optimal nonpseudorandom generator*. The proof of this is similar to the proof of Theorem 5.10 [NW94]. We start with some basic definitions.

A pseudorandom generator against a class of circuits is a function which takes a random seed as input and outputs a sequence of bits which is a pseudorandom distribution.

Definition 5.6 (Pseudorandom Generators). *A function G is said to be a $l(n)$-pseudorandom generator if*

1. *$G = \{G_n\}_{n>0}$ with $G_n : \Sigma^{l(n)} \to \Sigma^n$*
2. *G_n is computable in $2^{O(l(n))}$ time*
3. *For sufficiently large n, $G_n(U_{l(n)})$ is $(n^2, 1/n)$-pseudorandom.*

Definition 5.7 (Optimal Pseudorandom Generators). *A function G is said to be an optimal pseudorandom generator if it is an $O(\log n)$-pseudorandom generator.*

Nisan and Wigderson [NW94] showed that there is a connection between pseudorandom generators and *hard functions* in EXP:

Definition 5.8 (Hard Function). *A family of functions $f = \{f_n\}_{n\in\mathbb{N}}$ where $f_n : \Sigma^n \to \Sigma$ is (S, ϵ)-hard for any $S = S(n) > n$ and $\epsilon = \epsilon(n) > 0$, if for all large enough n, for all circuits C of size at most $O(S(n))$,*

$$U_n[C(x) = f_n(x)] \leq \frac{1}{2} + \epsilon(n).$$

The following theorem shows that under the assumption of existence of hard function in EXP, optimal pseudorandom generator exists [NW94].

Theorem 5.10 ([NW94])**.** *There exists an optimal pseudorandom generators if and only if there is a language L in EXP and $\exists\delta > 0$ such that L on inputs of length n is $(2^{\delta n}, 1/2^{\delta n})$-hard.*

The proof of the above theorem is constructive and thus we can explicitly convert optimal pseudorandom generators to the hard function and conversely. However this is still a very strong requirement and later Impagliazzo and Wigderson weakened it.

Theorem 5.11 ([IW96])**.** *Suppose there is a language L in EXP and $\exists\delta > 0$ such that L on inputs of length n cannot be solved by circuits of size at most $2^{\delta n}$. Then there exists a language L' in EXP and $\exists\delta' > 0$ such that L' on inputs of length n is $(2^{\delta' n}, 1/2^{\delta' n})$-hard and as a consequence optimal pseudorandom generator exists.*

Now let us state and prove the main result of this section.

Theorem 5.12. *Consider any $s \in (0,1]$ and $c > 0$. If there exists an algorithm $G = \{G_n\}_{n\in\mathbb{N}}$ where $G_n : \Sigma^{c\log n} \to \Sigma^n$ computable in $2^{O(\log n)}$ such that $\dim(\{G_n(U_{c\log n})\}) \geq s$, then P=BPP.*

Proof. Suppose $X := \{G_n(U_{c\log n})\}_{n\in\mathbb{N}}$. If $\dim(X) = s > 0$, then for all large enough n, there must be a subset of indices $S \subseteq \{1, 2, \cdots, n\}$ such that $|S| = \log n$ and for any $i \in S$, loss incurred by any polynomial-size predictor at i-th bit position is non-zero or in other words, for any polynomial-size circuit family $C = \{C_n\}_{n\in\mathbb{N}}$, $X[C_i(x_1, \cdots, x_{i-1}) = x_i] < 1$. Actually one can show a much stronger claim that there exists such a subset S such that $|S| = c \cdot n$, for some constant $c < 1$. Otherwise $\dim(X) = 0$. To prove this, suppose $|S| = o(n)$ and thus there exists a predictor π such that for every $w \in \Sigma^n$,

$$\mathrm{Loss}(\pi, w) = \sum_{i=1}^{n} \mathit{loss}(\pi(w[1 \ldots i-1],\ w[i])) = \sum_{i\in S} \mathit{loss}(\pi(w[1 \ldots i-1],\ w[i])).$$

Now as $\mathit{loss}(\pi(w[1 \ldots i-1],\ w[i])) \leq 1$, so for all large enough n, for any $\epsilon < \frac{1}{2}$, $\mathrm{LossRate}_\epsilon(\pi, X_n) = 0$. Hence $\mathrm{unpred}(X) = 0$ and by Corollary 3.2.1, $\dim(X) = 0$ as well.

Suppose S contains first $\log n$ many such indices. Also assume that $S = \{i_1, i_2, \cdots, i_{\log n}\}$ and $i_1 < i_2 < \cdots < i_{\log n}$. Now we define two languages L_0 and L_1 as follows: for $j = 0, 1$,

$$L_j := \{y \in \Sigma^{\log n - 1} | \exists x \in \Sigma^n \text{ in the support of } G_n \text{ and } x_S = jy\}$$

where jy denotes the string generated by concatenating j and y. First of all, note that as $i_1 \in S$, none of L_0 and L_1 is an empty set. Now clearly either L_0 or L_1 is the language that satisfies all the conditions of Theorem 5.11 [IW96]. Otherwise, there exists a predictor circuit of size at most $2^{\delta\log n}$, for some $\delta > 0$, i.e., polynomial in n, by which we can predict $i_{\log n}$-th bit position or loss incurred by that predictor at $i_{\log n}$-th bit position will be zero implying $i_{\log n} \notin S$ which is a contradiction. Thus either L_0 or L_1 can be used to construct an *optimal pseudorandom generator* and that eventually implies P=BPP. □

Chapter 6

Conclusion

In this part of this thesis, we study the question of quantifying amount of pseudorandomness present in a distribution. In case of randomness the information theoretic notion of min-entropy is widely used. The computational analogue of information theoretic notion of entropy, namely pseudoentropy is used in case of pseudorandomness. However, there are several different types of pseudoentropy, e.g., HILL-type pseudo entropy, Yao-type pseudo entropy, metric type pseudo entropy, next-bit pseudoentropy [HILL99, Yao82, HRV10, VZ12]. It is not at all clear whichever is the most appropriate. In this thesis, we propose an alternate notion by adapting the theory of dimension defined via betting functions s-gales [Lut03b] (Chapter 3). We compare our notion of dimension with different types of pseudoentropy (Chapter 4). We show a close relationship between dimension and next-bit pseudoentropy. However, we think that a much stronger relation can be established. In particular, Theorem 4.5 can be improved and the constant additive gap between two notions can be resolved.

After the complete characterization of distributions in terms of amount of pseudorandomness present in them, the next natural question that we address in this thesis is to extract out those pseudorandom part (Chapter 5). We show that deterministically we can not do that and also show $\Omega(\log n)$ lower bound on the number of extra pure random bits required. Unfortunately, we do not get significant success on constructing one such pseudorandom extractor. For the distributions with high HILL-type pseudoentropy it was previously known that any randomness extractor suffices to do this job [BSW03], but the class of distributions with high dimension is a strict superset of the class of distributions having high HILL-type pseudoentropy. So it is interesting to construct pseudorandom extractor for this much larger class of distributions. This is the main open problem that we pose as Question 5.2 in Chapter 5.

Before closing this part, we want to remind the reader about another thing that all the notions of pseudoentropy came while giving construction of pseudorandom generator from any one-way function. Till now the most efficient construction used the notion of next-bit pseudoentropy [VZ12]. We show that our notion of dimension is actually very closely related to next-bit pseudoentropy. So it is interesting to come up with a direct construction of pseudorandom generator from any one-way function using our notion of dimension and it is quite possible that our notion will lead to a much more efficient construction.

Part II

Time-space Trade-off in Small Space Computation

Chapter 7

Introduction

One of the central open question in the domain of space bounded computation is whether non-deterministic log-space class (NL) and deterministic log-space class (L) are equal or not. Another important but much easier question is whether NL is inside SC (simultaneous poly-logarithmic space and polynomial time) or not. On the other hand, it is already known that randomized log-space class (RL), which is a sub-class of NL is in SC [Nis95]. In this thesis we make a little step towards resolving this latter question.

7.1 The Reachability Problem

The graph reachability problem is defined as follows: Given a graph G and two vertices s and t in it, the problem is to decide whether there is a path from s to t in G. This problem is central to the field of complexity theory, particularly to the domain of space bounded computation. The graph reachability problem for different classes of graphs capture different complexity classes. For directed graphs this problem is complete for the complexity class NL and a celebrated result by Reingold showed that for undirected graphs the reachability problem is actually L-complete [Rei08]. Certain promise version of reachability problem for directed graphs is also known to be complete for the complexity class RL [RTV06]. Wigderson gave a fairly comprehensive survey that discusses the complexity of reachability in various computational models [Wig92].

Naturally for a long time the reachability problem has been the prime focus to the researchers of complexity theory. Clearly designing an algorithm that solves directed reachability problem and runs simultaneously in polynomial time and poly-logarithmic space will completely resolve the question "NL $\subseteq$ SC?" mentioned earlier. The most elementary graph traversal algorithms such as DFS (depth first search) and BFS (breadth first search) solve this problem in linear time, but need linear space. On the other hand, we have a $O(\log^2 n)$ space algorithm due to Savitch [Sav70], however it requires $O(n^{\log n})$ time. The main question is whether we can design an algorithm that is as efficient as (a little blow is also allowed) BFS/DFS in terms of time and as efficient as Savitch's algorithm in terms of space requirement. In his survey Wigderson asked whether it is possible to design a polynomial time algorithm that uses only $O(n^\epsilon)$ space, for some constant $\epsilon < 1$ [Wig92]. This question is also still open. In 1992, Barnes, Buss, Ruzzo and Schieber made some progress on this problem and gave an algorithm for directed reachability that requires polynomial time and $O(n/2^{c\sqrt{\log n}})$ space, for some constant c [BBRS92]. Till today this is the best known simultaneous time-space upper bound known for general directed reachability problem. Henceforth we refer to this bound as the BBRS bound. Improving the BBRS bound remains a significant open question regarding the complexity of the graph reachability problem.

In last five years there has been some progress on improving the BBRS bound for certain special subclasses of directed graphs. Planar graphs are a natural topological restriction of general graphs consisting of graphs that can be embedded on the surface of a plane such that no two edges cross. *Grid graphs* are a subclass of planar graphs, where the vertices are placed at the lattice points of a two dimensional grid and edges occur between a vertex and its immediate adjacent horizontal or vertical neighbor. Asano and Doerr provided a polynomial time algorithm to compute the *shortest path* between two given vertices (hence can decide reachability) in grid graphs which uses only $O(n^{1/2+\epsilon})$ space, for any constant $\epsilon > 0$ [AD11]. Later Imai *et al.* achieved a similar bound for reachability problem in planar graphs [INP$^+$13] by giving a space efficient construction of *separator* for planar graphs. Note that although it is known that reachability problem in grid graphs is log-space reducible to planar reachability, since this class (polynomial time and $O(n^{1/2+\epsilon})$ space) is not closed under log-space reductions, planar reachability does not follow from grid graph reachability. In [SV12], it was shown that the reachability problem for directed *acyclic* graphs with $O(n^{1-\epsilon})$ *sources nodes* and embedded on surfaces of $O(n^{1-\epsilon})$ genus can be solved in polynomial time and $O(n^{1-\epsilon})$ space. Recently Asano *et al.* gave a $\tilde{O}(\sqrt{n})$ space and polynomial time algorithm for reachability in planar graphs, thus improving upon the previous space bound [AKNW14]. We refer the reader to a survey article by Vinodchandran [Vin14] for more details on known results. In [CPT$^+$14] a $\tilde{O}(n^{2/3}g^{1/3})$ space and polynomial time bound was shown in case of graphs embedded on orientable surface of genus g.

In another line of work, Kannan *et al.* gave a $O(n^{\epsilon})$ space and polynomial time algorithm for deciding reachability in *unique path graphs* [KKR08]. Unique path graphs are a generalization of *strongly unambiguous* graphs and reachability problem in strongly unambiguous graphs is known to be in SC [BJLR91, Coo79]. Reachability in strongly unambiguous graphs can also be decided by a $O(\log^2 n/\log\log n)$ space algorithm, however this algorithm requires super polynomial time [AL98]. Interested readers may refer to a survey by Allender [All07] to further understand the results on the complexity of reachability problem in UL and on certain special subclasses of directed graphs.

There is some evidence that it might be hard to improve BBRS bound for general directed graphs by known techniques. The reason is that there are matching lower bounds known for deciding reachability in general directed graphs on certain restricted model of computation, namely NNJAG model [CR80, Poo93, EPA99] and all the algorithms known so far for the general reachability problem can be implemented in NNJAG without significant blow up in time and space.

7.1.1 Our Contribution on the Reachability Problem

Our first contribution of the Part II of this thesis is providing a new algorithm for solving reachability problem in case of H-minor-free graphs, where H is an arbitrary but a fixed graph. Our result improves upon the BBRS bound with an assumption that that along with the input graph we are also given the *tree decomposition* of the H-minor-free graph. Below is the formal statement of our result.

Theorem 7.1 (Theorem 9.1)**.** *Given a graph H, there is an algorithm that, given any H-minor-free graph G together with*

1. *a tree decomposition (T, X) of G, and*
2. *for every $X_i \in X$, the combinatorial embedding of the subgraph G_0 of $G[X_i]$,*

and two vertices s and t in G, decides whether there is a directed path from s to t in G. The algorithm runs in polynomial-time and uses $\tilde{O}(n^{2/3})$ space, where n is the number of vertices of the graph.

Details of the notations used in the statement of the above theorem can be found in Section 9.1.1 of Chapter 9 where we provide the proof of the above stated result. This key ingredient of the proof of the above theorem is designing a $\tilde{O}(n^{2/3})$-space and polynomial-time algorithm for constructing a 2/3-separator of size $O(n^{2/3})$ for the given graph. Once such a separator is obtained, we use ideas from [INP$^+$13] to design the reachability algorithm. To construct such a separator for H-minor-free graphs, we use the tree decomposition of the given graph by [RS03] and find a "separating node" in that tree. Then we construct a bounded-genus graph from the graph induced by the separating node. Finally by using the planarizing set construction used to establish time-space bound for reachability problem for high-genus graphs [CPT$^+$14], we design an algorithm to construct a planarizing set of size $O(n^{2/3})$ of the underlying undirected graph in polynomial-time and $\tilde{O}(n^{2/3})$ space.

For $K_{3,3}$-free and K_5-free graphs we give a better upper bound than the one given in Theorem 9.1. Kuratowski's theorem states that planar graphs are exactly those graphs that do not contain $K_{3,3}$ and K_5 as minors. Hence it is a natural question whether results on planar graphs can be extended to graphs that do not contain either a $K_{3,3}$ minor (known as $K_{3,3}$-free graphs) or a K_5 minor (known as K_5-free graphs). Certain complexity upper bounds that hold for planar graphs have been shown to hold for $K_{3,3}$-free and K_5-free graphs [BTV09, TW09, DLN$^+$09, DNTW09]. On the other hand, there are problems for which upper bounds that hold for planar graphs are not known to extend to such minor-free graphs (such as computing a perfect matching in bipartite graphs [MN95, TV12]). We show that the time-space bound known for planar graphs can also be obtained for both these classes of graphs. Here it is important to note that even though directed reachability in $K_{3,3}$-free and K_5-free graphs reduces to directed planar reachability[TW09], the reduction blows up the size of the graph by a polynomial factor and hence we cannot use this approach for our purposes.

Theorem 7.2 (Theorem 9.2)**.** *For any constant $0 < \epsilon < 1/2$, there is a polynomial time and $O(n^{1/2+\epsilon})$ space algorithm that given a directed $K_{3,3}$-free or K_5-free graph G on n vertices, decides whether there is a directed path from s to t in G.*

Although in case of H-minor-free graphs we need additional inputs like the tree decomposition and the embeddings of the bounded genus parts, in the above theorem we do not have any such requirements. The proof idea of the above stated theorem is similar to that of Theorem 7.1. However we use the known algorithm to compute a planar separator instead of a bounded genus separator. This gives better space bound compared to the case of H-minor-free graphs.

Next we show a polynomial time and $O(n^{\epsilon})$ space, for any constant $\epsilon > 0$ bound for deciding reachability in directed layered planar graphs. A layered planar graph is a planar graph where the vertex set is partitioned into layers and every edge occurs between two consecutive layers only. Our result significantly improves upon the previous space bound due to [INP$^+$13] and [AKNW14] for layered planar graphs.

Theorem 7.3 (Theorem 10.1)**.** *For every $\epsilon > 0$, there is a polynomial time and $O(n^\epsilon)$ space algorithm that decides reachability in directed layered planar graphs.*

Reachability in layered grid graphs (denoted as LGGR) is in UL which is a subclass of NL [ABC$^+$09]. Subsequently this result was extended to the class of all planar graphs [BTV09]. Allender et al also gave some hardness results for the reachability problem in certain subclasses of layered grid graphs. Specifically they showed that, 1LGGR is hard for NC^1 and 11LGGR is hard for TC^0 [ABC$^+$09]. Both these problems are however known to be contained in L though.

Firstly we argue that its enough to consider layered grid graphs (a subclass of general grid graphs). We divide a given layered grid graph into a courser grid structure along k horizontal and k vertical lines. We then design a modified DFS strategy that makes queries to the smaller graphs defined by these gridlines (we assume a solution in the smaller graphs by recursion) and visits every reachable vertex from a given start vertex. The modified DFS stores the highest visited vertex in each vertical line and the left most visited vertex in each horizontal line. We use this information to avoid visiting a vertex multiple number of times in our algorithm. We choose the number of horizontal and vertical lines to divide the graph appropriately to ensure that the algorithm runs in the required time and space bound. We refer the reader to Chapter 10 for the detailed proof.

7.2 Some Other Graph Theoretic Problems

In a given weighted directed graph computing shortest path between a pair of vertices is a fundamental problem in computer science. There are several popular and efficient algorithms that are known for this problem such as Dijkstra's algorithm [Dij59] and Bellman-Ford algorithm [Bel58, For56] (henceforth we refer it as the procedure `Bellman-Ford`). Both of these are polynomial time algorithms, but require linear amount of space. Among them Bellman-Ford algorithm is more versatile since it is also able to handle graphs with negative edge weights (but no negative weight cycles). There is also a more recent algorithm by Klein, Mozes and Weimann [KMW09] which runs in polynomial time (with better parameters) but still requires linear space. However this algorithm considers shortest path problem only for directed planar graphs.

The natural question is whether we can extend the results known for reachability problem to the shortest path problem. As mentioned in the earlier part of this chapter that for grid graphs, we already have a $O(n^{\frac{1}{2}+\epsilon})$ space and polynomial time algorithm for the shortest path problem due to [AD11]. In their paper, Asano and Doerr posed the question whether their result can be extended to planar graphs in general. In this thesis, we give a positive answer to their question and exhibit the first sub-linear space, polynomial time algorithm for the shortest path problem in planar graphs. Note that the shortest path problem for both undirected and directed graph is NL-complete [Gol08]. Before proceeding further we want to mention that in an independent work Asano *et al.* pointed out that their approach for solving reachability can also be extended to shortest path problem while achieving similar bound upto some polynomial blow up in time [AKNW14].

One interesting generalization of the reachability problem is the RedBluePath problem. Given a graph where all the edges are colored either red or blue and two vertices the problem is to decide the existence of a path between those vertices such that it alternates between red and blue colored edges. The RedBluePath problem is NL-complete even when restricted to planar directed acyclic graphs or in short planar DAGs [Kul11]. A natural relaxation of the above problem is EvenPath problem defined in Section 11.2. In general EvenPath problem is NP-complete [LP84], but for planar graphs it is known to be in P [Ned99]. In this thesis, we also give the first sublinear space and polynomial time algorithm for the RedBluePath and the EvenPath problem in case of planar DAGs.

Another central problem in the field of algorithms and complexity theory is the problem of finding the perfect matching, denoted as PerfectMatching. The best known upper bound for the PerfectMatching problem is *non-uniform* SPL [ARZ98] and the best hardness known is NL-hardness [CSV84]. However, for planar graphs the PerfectMatching problem is known to be L-hard [DKLM10]. It is also known that the PerfectMatching problem is in the complexity class RNC [KUW85, MVV87] and very recently it has been shown to be in quasi-NC in case of bipartite graphs [FGT16]. If we consider only the planar bipartite graphs, then the PerfectMatching problem is known to be even inside UL [DGKT12]. We know a polynomial time solution for the PerfectMatching problem in bipartite graphs using Ford-Fulkerson algorithm for network flow [KT05], but that takes the space linear in number of edges present in the graph. Unfortunately, no strictly sublinear ($O(n^{1-\epsilon})$, for some $\epsilon > 0$) space and polynomial time algorithm is known for the PerfectMatching problem even in case of planar bipartite graphs. Same is true for the problem of finding Hall-obstacle (denoted as HallObs (Decision + Construction) for planar bipartite graphs. In general, for any bipartite graph with two partitions of vertices A and B, Hall-obstacle is a subset of vertices of A whose neighboring set in B is of strictly smaller size. It is known from [DGKT12], that HallObs (Decision) is in co-UL and HallObs (Construction) is in NL, when the graph under consideration is planar bipartite.

The problem ExactPM (Decision) (first posed in [PY82]) denotes the problem of deciding the presence of a perfect matching in a given graph G with edges colored with red or blue and containing exactly k red colored edges for an integer k. This problem is not even known to be in P. Now we consider a natural relaxation of the above problem just by concentrating on the perfect matching containing even number of red edges and denote this problem as EvenPM. EvenPM problem is in P for bipartite graphs and in NL for planar bipartite graphs [DGKT12]. Till now, we do not know about any sublinear space and polynomial time algorithm for the EvenPM problem when concentrating only on planar bipartite graphs.

7.2.1 Our Contribution

In this thesis, we prove the following results.

Theorem 7.4 (Theorem 11.1)**.** *For directed planar graphs (containing no negative weight cycle and weights are bounded by polynomial in n) and for any constant $0 < \epsilon < \frac{1}{2}$, there is an algorithm that solves the* ShortestPath *problem in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space, where n is the number of vertices of the given graph.*

We use the space efficient construction of separator for planar graphs [INP$^+$13], and this is one of the main building blocks for the results stated in this section. Let the separator be S. Now calculate the shortest distance of every $v \in S$ from the vertex s. The shortest path from s to t must pass through the vertices in the separator and thus knowing the shortest path from s to each of such vertices is enough to find the shortest path from s to t. The shortest path from s to any $v \in S$ can be found by applying the same shortest path algorithm recursively to each of the connected component of the graph induced by the set of vertices $V \setminus S$. As a base case we use `Bellman-Ford` algorithm to find the shortest path. The recursive invocation of the above technique will lead to the time-space bound mentioned in the above theorem.

Another main contribution is to give an algorithm for the RedBluePath problem in planar DAGs. The main idea behind our algorithm is to use a modified version of DFS algorithm along with the planar separator.

Theorem 7.5 (Theorem 11.2)**.** *For any constant $0 < \epsilon < \frac{1}{2}$, there is a polynomial time algorithm that solves the* RedBluePath *problem for planar DAGs using $O(n^{\frac{1}{2}+\epsilon})$ space.*

Note that this is the first simultaneous $O(n^{1-\epsilon})$, for any $\epsilon > 0$ space and polynomial time bound known for any NL-complete problems. However as the corresponding complexity class is not closed under log-space reduction we do not get any containment result for the class NL.

Now using the reduction given in [Kul11] and the algorithm stated in the Theorem 7.5, we get an algorithm to solve the directed reachability problem for a fairly large class of graphs described in Section 11.2, that takes polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space. Thus we can able to beat the BBRS bound for such class of graphs.

We also establish a relation between the EvenPath problem in a planar DAG and the problem of finding odd length cycle in a directed planar graph and thus we argue that both of these problems have the same simultaneous time-space complexity. We use two colors red and blue to color the vertices of the given graph and then use the color assigned to the vertices of the separator to detect the odd length cycle. The conflicting assignment of color to the same vertex in the separator will lead to the presence of an odd length cycle. Here also we use the recursive approach to color the vertices and as a base case we use the well known BFS algorithm to solve the problem of detecting odd length cycle in each small component. Thus we get the following theorem regarding solving the EvenPath problem.

Theorem 7.6 (Theorem 11.3)**.** *For any constant $0 < \epsilon < \frac{1}{2}$, there is a polynomial time algorithm that solves the* EvenPath *problem in planar DAGs using $O(n^{\frac{1}{2}+\epsilon})$ space.*

Our another contribution is to give a time-space efficient algorithm for perfect matching problem in case of planar bipartite graphs.

Theorem 7.7 (Theorem 11.4)**.** *In planar bipartite graphs, for any constant $0 < \epsilon < \frac{1}{2}$,*

1. PerfectMatching *(Decision + Construction) can be solved in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space; and*
2. HallObs *(Decision + Construction) can be solved in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space.*

We build on the Miller and Naor's algorithm [MN95] for perfect matching in planar bipartite graphs. We show that this algorithm takes polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space as the only hard part of this algorithm is to find the shortest distance. We also argue that problem of finding Hall obstacle is directly associated with the problem of finding negative weight cycle and thus get the same simultaneous time-space bound for this problem as of the problem of detecting negative weight cycle.

Next we show that the complexity of even perfect matching in planar bipartite graphs is same as that of the perfect matching problem in planar bipartite graphs and deciding the presence of odd length cycle in directed planar graphs. Thus we get the following theorem for the EvenPM problem.

Theorem 7.8 (Theorem 11.5)**.** *For any constant $0 < \epsilon < \frac{1}{2}$, there exists an algorithm that solves the* EvenPM *problem for planar bipartite graphs in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space.*

7.3 Organization of Part II of the Thesis

The rest of the Part II of this thesis is organized as follows. In Chapter 8, we give some basic definitions and notations and also state certain earlier results that we use in this thesis. In Chapter 9, we present the algorithm for reachability in H-minor-free graphs and as a corollary we show Theorem 7.2. In Chapter 10, we give a proof of Theorem 7.3 related to deciding reachability in layered planar graphs. Finally in Chapter 11 we talk about simultaneous time-space bound of several important graph theoretic problems other than reachability for some specific graph classes. In particular, in Section 11.1 we give an algorithm for shortest path problem in directed planar graphs. In Section 11.2, we give a simultaneous time-space bound for deciding the presence of Red-Blue Path in a planar DAG. We also establish a relation between the problem of deciding the presence of an odd length cycle in directed planar graphs with the problem of deciding the presence of even path between two given vertices in planar DAGs and thus give the same simultaneous time-space bound for both of these problems. And finally in Section 11.3, we discuss the simultaneous time-space bound of some matching problems in planar bipartite graphs.

Chapter 8

Preliminaries

In this chapter, we give the basic definitions and known theorems used in this thesis. We first define some notations which will be used later in the subsequent chapters of this thesis.

8.1 Notations

In this thesis, we follow the standard model of computation to discuss the complexity measures of the stated algorithms. In particular, we consider the computational model in which an input appears on a read-only tape and the output is produced on a write-only tape and we only consider the internal read-write tape in the measure of space complexity. For the basic definitions of the complexity classes used in this thesis, we refer the reader to any standard book on complexity theory (e.g., [AB09]).

Throughout this thesis, by log we mean logarithm to the base 2. We denote the set $\{1, 2, \cdots, n\}$ by $[n]$. In this thesis, by $\widetilde{O}(s(n))$ we mean $O(s(n)(\log n)^{O(1)})$. A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E where each edge can be represented as an ordered pair (u, v) in case of directed graph and as an unordered pair $\{u, v\}$ in case of undirected graph, such that $u, v \in V$. A *directed acyclic graph* is a directed graph containing no directed cycle. Unless it is specified, in this thesis G will denote the directed graph, where $|V| = n$. Given a graph G, let $V(G)$ and $E(G)$ denote the set of vertices and the set of edges of G respectively. For a set of vertices X, let $G[X]$ denote the subgraph of G induced by X. Given a directed graph G, we denote the underlying undirected graph by $\hat{G}$. For the basic definitions and terms of graph theory used in this thesis, interested readers may refer to any standard book on graph theory (e.g., [Die12]).

8.2 Graph Embedding and Planarity

Now we define necessary notions on graphs embedded on surfaces. We refer the reader to the excellent book by Mohar and Thomassen [MT01] for a comprehensive treatment of this topic. In this thesis we only consider *closed orientable* surfaces. These surfaces are obtained by adding "handles" to a sphere.

Let $G = (V, E)$ be a graph and for each $v \in V$, let π_v be a cyclic permutation of edges incident on v. Let $\Pi = \{\pi_v \mid v \in V\}$. We say that Π is a *combinatorial embedding* of G. Given a combinatorial embedding we can define a Π-*facial walk*. Let $e = \langle v_1 v_2 \rangle$ be an edge. Consider the closed[1] walk $f = v_1e_1v_2e_2v_3 \cdots v_ke_kv_1$ where $\pi_{v_{i+1}}(e_i) = e_{i+1}$, and $\pi_{v_1}(e_k) = e_1$. We call f a *face* of the graph G.

Given a Π-embedding of a graph G, the Π-*genus* of G is the unique g such that $|V| - |E| + |F| = 2 - 2g$, where F is the set of all faces of G. This is popularly known as the *Euler-Poincaré formula.*

It is known that given any graph with Π-genus g, it can be embedded on a closed orientable surface of genus g such that every face is homeomorphic to an open disc. Let Π be a combinatorial embedding of a graph G and H be a subgraph of G. The embedding Π naturally induces an embedding Π' on $G \setminus H$. By abuse of notation, we still refer to the induced embedding as Π-embedding.

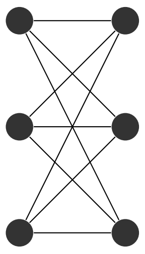
Planar graphs are the graphs that can be embedded on a surface of genus 0 (e.g., plane).

Definition 8.1 (Planar Graphs)**.** *A planar graph is a graph that can be drawn on a plane without any edge crossing.*

Wagner gave an alternate definition of planar graphs in terms of graphs with excluded minors [Wag37]. A graph H is said to be a *minor* of another graph G if one can obtain H from G by deleting some vertices and edges, and contracting some of the edges of G. G is called H-minor-free if H is not a minor of the graph G.

Theorem 8.1 ([Wag37])**.** *A graph is planar if and only if it is both $K_{3,3}$-free and K_5-free.*

$K_{3,3}$ is the complete bipartite graph where both the partitions of vertices are of size 3 and K_5 is the complete graph containing 5 vertices (Figure 8.1). These two graphs are examples of non-planar graphs and thus if are minor of some graph then that graph cannot be drawn on a plane without any edge crossing. Wagner's main contribution was to show the converse. A natural extension of planar graphs are the graphs that contain either one of $K_{3,3}$ or K_5, but not the both as minor. Such a graph would be a $K_{3,3}$-free graph or K_5-free graph.


(a) $K_{3,3}$

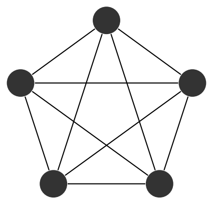

(b) K_5

Figure 8.1: The graphs $K_{3,3}$ and K_5

[1]A priori it is not obvious that that this leads to a closed walk. However, it can shown that this walk comes back to v_1. See [MT01] Chapter 3.2 for a proof.

8.3 Separator and Directed Planar Reachability

The notions of a planarizing set and a separator defined below are crucial in this thesis. A set S of vertices of a graph G is called a *planarizing set* if $G \setminus S$ is a planar graph. An (α, β)-*separator* of an undirected graph $G = (V, E)$ having n vertices, is a subset S of V such that $|S| \leq O(\alpha)$ and every connected component in $V \setminus S$ has at most βn vertices.

Next we state two theorems about planar graphs that are used in this thesis. In [INP$^+$13] the authors construct a $(n^{1/2}, 8/9)$-separator for a given undirected planar graph. By running their algorithm repeatedly (a constant number of times), we can obtain a $(n^{1/2}, 1/3)$ separator for a given undirected planar graph.

Theorem 8.2 ([INP$^+$13])**.** *Given an undirected planar graph G there is an algorithm that computes a $(n^{1/2}, 1/3)$-separator of G in polynomial time and $\tilde{O}(n^{1/2})$ space.*

We refer to the algorithm of this theorem as `PlanarSeparator` algorithm. By invoking this algorithm recursively, one can get an $(n^{\frac{1}{2}+\epsilon}, n^{-\epsilon})$-separator for any $0 < \epsilon < 1/2$, in polynomial time and $\widetilde{O}(n^{\frac{1}{2}+\epsilon})$ space. We refer the algorithm for getting such $(n^{\frac{1}{2}+\epsilon}, n^{-\epsilon})$-separator as `PlanarSeparatorFamily` algorithm. In [INP$^+$13], this algorithm is used to obtain a time-space efficient algorithm for reachability on directed planar graphs.

Theorem 8.3 ([INP$^+$13])**.** *For any constant $0 < \epsilon < 1/2$, there is an algorithm that, given a directed planar graph G and two vertices s and t, decides whether there is a path from s to t in G. This algorithm runs in time $n^{O(1/\epsilon)}$ and uses $O(n^{1/2+\epsilon})$ space, where n is the number of vertices of G.*

We refer the algorithm mentioned in the statement of the above theorem as `PlanarReach` algorithm.

8.4 A Reachability Algorithm for High-genus Graphs

In [CPT$^+$14] we provide a space efficient algorithm to construct a planarizing set for an undirected graph embedded on an orientable surface. We restate the result here because we need it in establishing the main result of Chapter 9.

Theorem 8.4 ([CPT$^+$14])**.** *There is an algorithm that given a combinatorial embedding of an undirected graph G embedded on an orientable surface of genus g, outputs a planarizing set of G of size $O(n^{2/3}g^{1/3})$. This algorithm runs in polynomial time and uses space $\widetilde{O}(n^{2/3}g^{1/3})$, where n denotes the number of vertices of G.*

Now using the above construction of planarizing set we can solve the reachability problem for directed graphs embedded on an orientable surface. We restate the result below, however we do not use this result in this thesis.

Theorem 8.5 ([CPT$^+$14])**.** *There is an algorithm that, given a directed graph G embedded on an orientable surface of genus g with the combinatorial embedding and two vertices s and t, decides whether there is a path from s to t in G. This algorithm runs in polynomial-time and uses $\tilde{O}(n^{2/3}g^{1/3})$ space, where n is the number of vertices of the graph.*

Chapter 9

New Time-Space Upperbounds for Directed Reachability in H-minor-free Graphs

In this chapter, we describe a new reachability algorithm for H-minor-free graphs, where H is an arbitrary but fixed graph. Our algorithm improves upon the BBRS bound under some assumptions on input. Let us recall the formal statement of the result, i.e., Theorem 7.1.

Theorem 9.1. *Given a graph H, there is an algorithm that, given any H-minor-free graph G together with*

1. *a tree decomposition (T, X) of G, and*
2. *for every $X_i \in X$, the combinatorial embedding of the subgraph G_0 of $G[X_i]$,*

and two vertices s and t in G, decides whether there is a directed path from s to t in G. The algorithm runs in polynomial-time and uses $\tilde{O}(n^{2/3})$ space, where n is the number of vertices of the graph.

The reader may refer to Section 9.1.1 to understand the notation that we use in Theorem 9.1.

In case of $K_{3,3}$-free and K_5-free graphs, we give a much better upper bound than that given for general H-minor-free graphs. Note that $K_{3,3}$-free and K_5-free graphs are strict superset of planar graphs as due to Kuratowski's theorem, planar graphs are exactly those graphs which contain neither $K_{3,3}$ nor K_5 as minors. Hence it is a quite natural to ask the question whether the bounds known for planar graphs can also be achieved for such larger class of graphs. We provide a positive answer to this question in the following theorem which is a restatement of Theorem 7.2.

Theorem 9.2. *For any constant $0 < \epsilon < 1/2$, there is a polynomial time and $O(n^{1/2+\epsilon})$ space algorithm that given a directed $K_{3,3}$-free or K_5-free graph G on n vertices, decides whether there is a directed path from s to t in G.*

Although for Theorem 9.1 we require additional inputs (such as the tree decomposition and the embeddings of the bounded genus parts), in Theorem 9.2 we do not have any such requirements.

9.1 A Reachability Algorithm for H-minor-free Graphs

In this section, we prove Theorem 9.1 by first giving an algorithm to construct a separator of the input graph. Towards this we define the notion of a tree decomposition of a graph which is crucial to the construction.

9.1.1 Graph Minor Decomposition Theorem

Let us recall that a graph H is said to be a *minor* of a graph G if H can be obtained from a subgraph of G by contracting some edges. A graph G is said to be H*-minor-free* if G does not contain H as a minor, for some graph H.

Definition 9.1. *A* tree decomposition *of a graph $G = (V, E)$ is a tuple (T, X) where $T = (V_T, E_T)$ is a tree and $X = \{X_i \mid i \in V_T\}$ such that,*

1. $\cup_i X_i = V$,

2. for every edge (u, v) in G, there exists i, such that u and v belong to X_i, and

3. for every $v \in V$, the set of nodes $\{i \in V_T \mid v \in X_i\}$ forms a connected subtree of T.

We will refer to the X_i's as *bags* of vertices. Note that each bag corresponds to a node (we call vertices of T as *nodes*) in the tree T. The *width* of a tree decomposition (T, X), is the maximum over the size of X_i's minus 1. The *treewidth* of a graph is the minimum width over all possible tree decompositions of G. A tree decomposition is said to be a path decomposition if $T = (V_T, E_T)$ is a path and *pathwidth* of a graph is the minimum width over all possible path decompositions of G.

For a fixed graph H, Robertson and Seymour gave a tree decomposition for every H-minor-free graph [RS03]. Before we see this tree decomposition theorem we need to give a few definitions. First we define *h-almost embeddable* graphs. Before going into the actual definition, let us give some informal description. One can think "almost embeddable" graphs as bounded-genus graphs which have a small number of "local areas of non-planarity", called vortices, and a few vertices, called apices, having any number of incident edges that are not properly embedded. Formally, a graph G is called *h-almost embeddable* if there exists a set of vertices Y (called the *apices*) of size at most h such that,

i. $G \setminus Y$ can be written as $G_0 \cup G_1 \cup \ldots \cup G_h$,

ii. G_0 has an embedding on a surface (say $\mathcal{S}$) of genus at most h,

iii. for $i = 1, \cdots, h$, G_i's are pairwise disjoint (we shall refer to them as *vortices*),

iv. there exist faces $F_1, \cdots, F_h$ of G_0 and pairwise disjoint disks[1] $D_1, \cdots, D_h$ on $\mathcal{S}$ such that for all $i \in \{1, \ldots, h\}$, $D_i \subseteq F_i$ and $U_i := V(G_0) \cap V(G_i) = V(G_0) \cap D_i$, and

v. for each graph G_i, there is a path decomposition $(\mathcal{P}_u)_{u \in U_i}$ of width at most h such that $u \in \mathcal{P}_u$, for all $u \in U_i$. The sets of vertices in $\mathcal{P}_u$ are ordered according to the ordering of the corresponding u's as vertices along the boundary of face F_i in G_0.

[1] *Disk* is a two dimensional manifold with boundary and its boundary is a circle. Disks in surface $\mathcal{S}$ are also defined in the natural way.

Let G and H be two graphs each containing cliques of equal sizes. The *clique-sum* of G and H is formed by identifying pairs of vertices in these two cliques to form a single shared clique, and then possibly deleting some of the clique edges (may be none). A *k-clique-sum* is a clique-sum in which both cliques have at most k vertices. The k-clique-sum of G and H is denoted as $G \oplus_k H$. The set of shared vertices in this operation is called the *join set.*

We are now ready to state the decomposition theorem for H-minor-free graphs.

Theorem 9.3 ([RS03]). *For every graph H, depending only on $|V(H)|$, there exists an integer $h \geq 0$ such that every H-minor-free graph can be represented as at most h-clique-sum of "h-almost embeddable" graphs in some surface on which H cannot be embedded.*

Henceforth, we will assume that the tree decomposition of the original graph and the combinatorial embedding of all subgraphs (the G_0's in each h-almost embeddable graph) that are embedded on the surface are provided as part of the input. We will refer to this as *tree decomposition with combinatorial embedding* of H-minor-free graphs. Note that Robertson and Seymour's proof gave us an $O(n^3)$ time algorithm for finding such decomposition which was later simplified in [KW11], but no log-space algorithm is known for this purpose. However, for some special subclasses like $K_{3,3}$-free or K_5-free graphs such log-space algorithm is known and we will discuss that in the latter part of this chapter. If one can come up with a log-space or even $\widetilde{O}(n^{2/3})$-space algorithm for finding above mentioned tree decomposition for H-minor-free graphs then we will no longer need the extra assumption on the input for our algorithm to work.

9.1.2 Constructing Separator for H-minor-free Graphs

We will show that given a decomposition of a H-minor-free graph stated in the last subsection, how to construct a separator. We start with the following lemma.

Lemma 9.1.1. *There exists a log-space algorithm, that given a tree decomposition (T, X) of an undirected graph G on n vertices, outputs a node $i \in T$ such that every connected component in $G[V \setminus X_i]$ has at most $n/2$ vertices.*

Proof. Pick a node $i \in T$. We shall refer to i as the current node. If every connected component in $G[V \setminus X_i]$ has at most $n/2$ vertices then output i and stop. Otherwise, let C be the connected component in $G[V \setminus X_i]$ such that $|C| > n/2$. Let j be the unique neighbor of i in T such that $X_j \cap C \neq \emptyset$. The reason why j is unique is because, if there are more than one neighbors of i (say j_1 and j_2) in T, such that $X_{j_1} \cap C \neq \emptyset$ and $X_{j_2} \cap C \neq \emptyset$, then j_1 and j_2 are connected by a path in $T \setminus \{i\}$, since C is a subgraph of $G[V \setminus X_i]$. Now this path together with the edges (i, j_1) and (i, j_2) forms a cycle in T, which is a contradiction.

Now the sum of the number of vertices of all other connected components of $G[V \setminus X_i]$ other than C, together with X_i is less than $n/2$. Therefore, for the node $j \in T$, the largest connected component in $G[V \setminus X_j]$ has strictly lesser number of vertices than C. Now we set j as the current node and repeat the above process. The process terminates since $|C|$ that we obtain in each step, strictly decreases. $\square$

We now give a separator construction for all H-minor-free graphs which is the main contribution of this section.

Theorem 9.4. *Given an undirected H-minor-free graph G and its tree decomposition with combinatorial embedding, there exists an $\tilde{O}(n^{2/3})$ space, polynomial time algorithm that computes a $(n^{2/3}, 2/3)$-separator of G.*

Proof. Given an input graph G and its tree decomposition, compute the vertex i using Lemma 9.1.1. The separator for G that we would construct would be a subset of X_i.

Now if $|X_i| \leq O(n^{2/3})$, then it follows from Lemma 9.1.1 that X_i is a $(n^{2/3}, 1/2)$-separator of G. Otherwise, consider the node i and its corresponding h-almost embeddable graph $K = G[X_i]$. Now consider the representation of K using apices and vortices. Let Y be the set of apices and $K \setminus Y$ can be written as $K_0 \cup K_1 \cup \cdots \cup K_h$ where each of K_i has a path decomposition $(\mathcal{P}_u)_{u \in U_i}$ of width less than h. Now build a new graph K' from K_0 using the following steps: for $i = 1, \cdots, h$, add a cycle of length $|\mathcal{P}_u|$ attached to the vertex $u \in U_i$ inside the face F_i and then connect those cycles such that they form a path like structure similar to the corresponding path decomposition. The new graph K' is a graph embedded on a constant genus and so from Theorem 8.4, we can get a $(n^{2/3}, 2/3)$-separator S (which is union of planarizing set of K', say Z and output of `PlanarSeparator` on the graph $K' \setminus Z$) using $\tilde{O}(n^{2/3})$ space and polynomial time. If S contains some vertices from a newly added cycle, then we add all the vertices present in the corresponding "bag" of vertices of the respective path decomposition. We also add all the apices of K_0 and we get a new set S'. As the size of S is $O(n^{2/3})$, so the size of S' will be at most $O(hn^{2/3}) = O(n^{2/3})$.

Claim 9.1.1. *S' is a $(n^{2/3}, 2/3)$-separator of K.*

Proof. Observe that by construction, K' is a graph embedded on a bounded genus surface. Moreover there is a canonical injective map (say σ) from vertices in K to vertices in K'. To see this, note that $K' = K_0 \cup$ newly added cycles and by construction, for every vertex in the bag X_i there is a vertex in the newly added cycle in K'.

Since S is a $(n^{2/3}, 2/3)$-separator of K', S' is also a $(n^{2/3}, 2/3)$-separator of K. Let C be a connected component in $K \setminus S'$. Then the vertices corresponding to C in K' (via the map σ) also forms a connected component. Since every connected component in $K' \setminus S$ has size at most $2|K'|/3$, so S' is a $(n^{2/3}, 2/3)$-separator of K. □

By running the above construction repeatedly (a constant number of times), we can get a $(n^{2/3}, 1/6)$-separator $\overline{S}$. As according to Lemma 9.1.1, $G[V \setminus X_i]$ contains at most $n/2$ vertices, so the set $\overline{S}$ also acts as a $(n^{2/3}, 2/3)$-separator for the whole graph G. It is clear from the construction of $\overline{S}$ that this algorithm will take $\tilde{O}(n^{2/3})$ space and polynomial time. □

We also consider the special case when H is either the $K_{3,3}$ or the K_5.

Theorem 9.5 ([Wag37, TW09]). *Let (T, X) be a tree decomposition of a $K_{3,3}$-free or K_5-free graph G. Then*

1. *for every $X_i \in X$, $G[X_i]$ is either a planar graph or the K_5 (if G is $K_{3,3}$-free) or V_8 (if G is K_5-free) (see Figure 9.1), and*

2. *G is the 3-clique-sum of $G[X_i]$ and $G[X_j]$ for every adjacent vertices i, j in T.*

Moreover given a $K_{3,3}$-free or K_5-free graph G, such a tree decomposition can be computed in log-space.

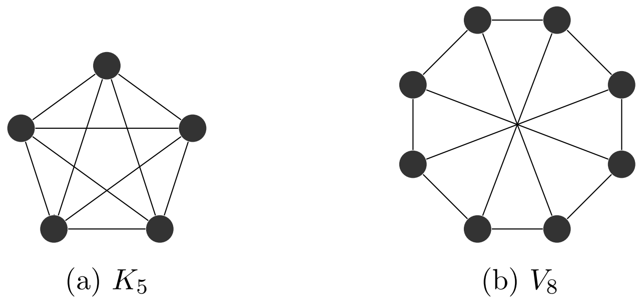


Figure 9.1: The graphs K_5 and V_8 (also known as Wagner's graph)

Thierauf and Wagner have shown how to compute the tree decomposition of a $K_{3,3}$-free or K_5-free graph given in Theorem 9.5 in log-space [TW09] and thus we get the following corollary for these special class of H-minor-free graphs.

Corollary 9.1.1. *Given an undirected $K_{3,3}$-free or K_5-free graph G, there exists an $\tilde{O}(n^{1/2})$ space, polynomial time algorithm that computes a $(n^{1/2}, 2/3)$-separator of G.*

Proof. Given an input graph G, first compute a decomposition tree T and compute the vertex i by running the algorithm from Lemma 9.1.1. The separator for G that we would construct would be a subset of X_i. Let i have m neighbors in T, say $i_1, \ldots, i_m$. Now for every j, $G[X_i]$ is joined with $G[X_{i_j}]$ using the clique-sum operation of at most 3 vertices. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ where C_j is a set of at most 3 vertices in X_i, such that $G[X_i]$ is joined with $G[X_{i_j}]$ via C_j.

Now if $|X_i| \le n^{1/2}$, then it follows from Lemma 9.1.1 that X_i is a $(n^{1/2}, 1/2)$-separator of G. Otherwise, we know from Theorem 9.5 that $G[X_i]$ is a planar graph. Now let S be a $(n^{1/2}, 1/3)$-separator of $G[X_i]$ as obtained by `PlanarSeparator`. By running the above construction repeatedly (a constant number of times), we can get a $(n^{1/2}, 1/6)$-separator $\overline{S}$.

Now define
$$\mathcal{P} = \{j \in [m] \mid C_j \cap \overline{S} \neq \emptyset\}.$$
Clearly, $|\mathcal{P}| \le |\overline{S}|$.

Let
$$S' = \overline{S} \cup \left(\cup_{j \in \mathcal{P}} C_j\right),$$
that is, S' is the separator $\overline{S}$ together with those sets C_j's such that each of them shares at least one vertex with $\overline{S}$. Since each C_j has size at most 3, $|S'| \le 3 \cdot |\overline{S}|$.

As according to Lemma 9.1.1, $G[V \setminus X_i]$ contains at most $n/2$ vertices, so the set S' also acts as a $(n^{1/2}, 2/3)$-separator for the whole graph G. Once we have a tree decomposition, it is easy to see that the set S' can be computed by a log-space algorithm with a constant number of oracle query to `PlanarSeparator` and hence we get the desired time and space bound. □

Proof of Theorem 9.1. Observe that the planar reachability algorithm of Theorem 8.3 essentially uses the properties that

i. a subgraph of a planar graph is also planar, and

ii. there exists an algorithm that computes a $(n^{1/2}, 2/3)$-separator of a planar graph in polynomial time and $\tilde{O}(n^{1/2})$ space.

Note that by the definition itself, all the subgraphs of a H-minor-free graph is also H-minor-free and given a tree decomposition, from Theorem 9.4 we get an algorithm that computes a $(n^{2/3}, 2/3)$-separator of a H-minor-free graph in polynomial time and $\tilde{O}(n^{2/3})$ space. Now using the algorithm stated in Theorem 8.3, we get our desired result. □

Proof of Theorem 9.2. By Corollary 9.1.1, given a $K_{3,3}$-free or K_5-free graph G, we can compute in $\tilde{O}(n^{1/2})$ space and polynomial time, a $(n^{1/2}, 2/3)$-separator of G. Once a separator is computed we can use a recursive method identical to the proof of Theorem 8.3 from [INP^{+}13] to design an $O(n^{1/2+\epsilon})$-space and polynomial time algorithm for the reachability problem. □

Chapter 10

An $O(n^\epsilon)$ Space and Polynomial Time Algorithm for Reachability in Directed Layered Planar Graphs

In this chapter, we show that reachability in directed layered planar graphs can be decided in polynomial time and $O(n^\epsilon)$ space for any constant $\epsilon > 0$. A layered planar graph is a planar graph where the vertex set is partitioned into layers (say L_0 to L_m) and every edge occurs between layers L_i and L_{i+1} only. Our result significantly improves upon the previous space bound due to [INP$^+$13] and [AKNW14] for layered planar graphs.

Theorem 10.1. *For every $\epsilon > 0$, there is a polynomial time and $O(n^\epsilon)$ space algorithm that decides reachability in directed layered planar graphs.*

Above theorem is the restatement of Theorem 7.3 mentioned in the introductory chapter of this part of the thesis. As a consequence of our result, it is easy to achieve the same time-space upper-bound for the reachability problem in *upward planar graphs*. We say that a graph is upward planar if it admits an upward planar drawing, i.e., a planar drawing where the curve representing each edge should have the property that every horizontal line intersects it in at most one point or in other words, every edge is monotonically non-decreasing in the y-direction on the plane. In the domain of graph drawing, it is an important topic to study the upward planar drawing of planar DAGs [BT87, BLR90]. It is NP-complete to determine whether a planar DAG with multiple sources and sinks has an upward planar drawing [GT95]. However, given an upward planar drawing of a planar DAG, the reachability problem can easily be reduced to reachability in a layered planar graph using only logarithmic amount of space and thus admits the same time-space upper bound as of layered planar graphs.

10.1 Class nSC and its Properties

$\mathsf{TISP}(t(n), s(n))$ denotes the class of languages decided by a deterministic Turing machine that runs in time $O(t(n))$ and uses $O(s(n))$ space. Then, $\mathsf{SC} = \mathsf{TISP}(n^{O(1)}, (\log n)^{O(1)})$.

Expanding the class SC, we define the complexity class nSC (short for `near-SC`) in the following definition.

Definition 10.1 (Complexity Class `near-SC` or nSC)**.** *For a fixed $\epsilon > 0$, we define $\mathsf{nSC}_\epsilon := \mathsf{TISP}(n^{O(1)}, n^\epsilon)$. The complexity class nSC is defined as*

$$\mathsf{nSC} := \bigcap_{\epsilon>0} \mathsf{nSC}_\epsilon.$$

We next show that nSC is closed under log-space reductions. This is an important property of the class nSC and will be used to prove Theorem 10.1. Although the proof is quite standard, but for the sake of completeness we provide it here.

Theorem 10.2. *If $A \leq_l B$ and $B \in \mathsf{nSC}$, then $A \in \mathsf{nSC}$.*

Proof. Let us consider that a log-space computable function f be the reduction from A to B. It is clear that for any $x \in A$ such that $|x| = n$, $|f(x)| \leq n^c$, for some constant $c > 0$. We can think that after applying the reduction, $f(x)$ appears in a separate write-once output tape and then we can solve $f(x)$, which is an instance of the language B and now the input length is at most n^c. Now take any $\epsilon > 0$ and consider $\epsilon' = \frac{\epsilon}{c} > 0$. $B \in \mathsf{nSC}$ implies that $B \in \mathsf{nSC}_{\epsilon'}$ and as a consequence, $A \in \mathsf{nSC}_\epsilon$. This completes the proof. $\square$

Now we will see if instead of deterministic Turing machine, we consider deterministic auxiliary pushdown machine, then what will be power of the corresponding complexity class. However, note that the result that we are going to discuss now is not required to prove our main theorem, i.e., Theorem 10.1 and thus is of independent interest. First we define the complexity class $\mathsf{P} - \mathsf{nSC}$ (short for `Pushdown near-SC`) as follows.

Definition 10.2 (Complexity Class `Pushdown near-SC` or $\mathsf{P} - \mathsf{nSC}$)**.** *For a fixed $\epsilon > 0$, we define $\mathsf{P} - \mathsf{nSC}_\epsilon$ to be the class of languages decided by a deterministic auxiliary pushdown machine that runs in time $n^{O(1)}$ and uses $O(n^\epsilon)$ space. The complexity class $\mathsf{P} - \mathsf{nSC}$ is defined as*

$$\mathsf{P} - \mathsf{nSC} := \bigcap_{\epsilon>0} \mathsf{P} - \mathsf{nSC}_\epsilon.$$

Next, we show that in the scenario we are concerned about, a deterministic auxiliary pushdown machine does not provide any extra power over a deterministic Turing machine.

Theorem 10.3. $\mathsf{P} - \mathsf{nSC}$=$\mathsf{nSC}$.

The above theorem comes as a corollary of an old result by Cook [Coo79] and here we first restate that result.

Theorem 10.4 ([Coo79])**.** *If a language is decided by a deterministic auxiliary pushdown machine that runs in time $t(n)$ and uses $s(n) \geq \log n$ space, then that language is in $TISP((t(n))^6, (s(n) + \log t(n)) \log t(n))$.*

Now it is easy to see that Theorem 10.3 follows from the above theorem.

Proof of Theorem 10.3. From the definition of deterministic auxiliary pushdown machine, it is trivial to see that for any $\epsilon > 0$, $\mathsf{nSC}_\epsilon \subseteq \mathsf{P} - \mathsf{nSC}_\epsilon$ and thus $\mathsf{nSC} \subseteq \mathsf{P} - \mathsf{nSC}$.

Now for the converse direction, let us consider a language $L \in \mathsf{P} - \mathsf{nSC}$ which implies $L \in \mathsf{P} - \mathsf{nSC}_\epsilon$, for any $\epsilon > 0$. Now by Theorem 10.4, $L \in \mathsf{nSC}_{2\epsilon}$. As a consequence, $L \in \mathsf{nSC}$ and this completes the proof. $\square$

10.2 Reachability in Layered Planar Graphs

In this section we prove Theorem 10.1. Let us start with the definitions of layered planar graph and layered grid graph.

Definition 10.3 (Layered Planar Graph)**.** *A planar graph $G = (V, E)$ is referred to as* layered planar *if it is possible to represent V as a union of disjoint partitions, $V = V_1 \cup V_2 \cup \cdots \cup V_k$, for some $k > 0$, and for any two consecutive partitions V_i and V_{i+1}, there is a planar embedding of edges from the vertices of V_i to that of V_{i+1} and there is no edge between two vertices of non-consecutive partitions.*

Now let us define the notion of layered grid graph and also note that grid graphs are by definition planar.

Definition 10.4 (Layered Grid Graph)**.** *A directed graph G is said to be a $n \times n$* grid graph *if it can be drawn on a square grid of size $n \times n$ and two vertices are neighbors if their L_1-distance is one. In a grid graph a edge can have four possible directions, i.e., north, south, east and west, but if we are allowed to have only two directions north and east, then we call it a* layered grid graph*.*

We use the following result of Allender *et al.* [ABC$^+$09] to simplify our proof.

Proposition 10.2.1 ([ABC$^+$09])**.** *Reachability problem in directed layered planar graphs is log-space reducible to the reachability problem in layered grid graphs.*

We show that the reachability problem in layered grid graphs (denoted as LGGR) is in nSC (Theorem 10.5). Then by applying Proposition 10.2.1 and Theorem 10.2 we have the proof of Theorem 10.1. So the main task remains is to show that LGGR is in nSC.

Theorem 10.5. LGGR $\in$ nSC*.*

To establish Theorem 10.5 we define an auxiliary graph in Section 10.2.1 and give the required algorithm in Section 10.2.2.

10.2.1 The Auxiliary Graph H

Let G be a $n \times n$ layered grid graph. We denote the vertices in G as (i, j), where $0 \leq i, j \leq n$. Without loss of generality, we can assume that $s = (0, 0)$ and $t = (n, n)$; otherwise instead of G, we consider the subgraph of G such that s be the leftmost and bottommost vertex and t be the rightmost and topmost vertex of that subgraph and follow the same algorithm. Let k be a parameter that determines the number of pieces in which we divide G. We will fix the value of k later to optimize the time and space bounds. Assume without loss of generality that k divides n. Given G we construct an auxiliary graph H as described below.

Divide G into k^2 many *blocks* (will be defined shortly) of size $n/k \times n/k$. More formally, the vertex set of H is

$$V(H) := \{(i, j) \mid i \text{ or } j \text{ is a non-negative multiple of } n/k.\}$$

Note that $V(H) \subseteq V(G)$. We consider k^2 many blocks $G_1, G_2, \cdots, G_{k^2}$, where a vertex $(i, j) \in V(G_l)$ if and only if $i'\frac{n}{k} \leq i \leq (i'+1)\frac{n}{k}$ and $j'\frac{n}{k} \leq j \leq (j'+1)\frac{n}{k}$, for some integer

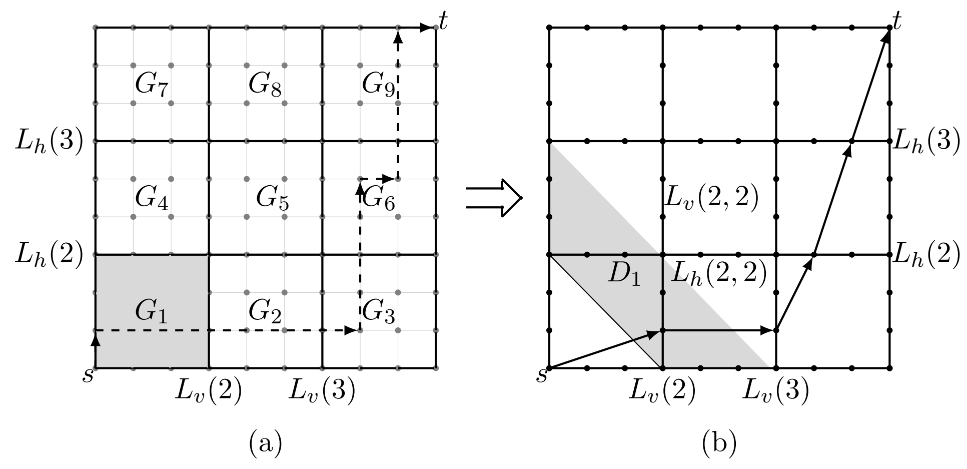


Figure 10.1: (a) An example of layered grid graph G and its decomposition into blocks (b) Corresponding auxiliary graph H

$i' \geq 0$ and $j' \geq 0$ and the vertices for which any of the four inequalities becomes equality, will be referred as *boundary vertices*. Moreover, we have $l = i' \cdot k + j' + 1$. $E(G_l)$ is the set of edges in G induced by the vertex set $V(G_l)$.

For every $i \in [k+1]$, let $L_h(i)$ and $L_v(i)$ denote the set of vertices, $L_h(i) := \{(i', j') | j' = (i-1)\frac{n}{k}\}$ and $L_v(i) := \{(i', j') | i' = (i-1)\frac{n}{k}\}$. When it is clear from the context, we will also use $L_h(i)$ and $L_v(i)$ to refer to the corresponding gridline in H. Observe that H has $k+1$ vertical gridlines and $k+1$ horizontal gridlines.

For every pair of vertices $u, v \in V(G_l) \cap V(H)$ for some l, add the edge (u, v) to $E(H)$ if and only if there is a path from u to v in G_l, unless $u, v \in L_v(i)$ or $u, v \in L_h(i)$ for some i. Also for every pair of vertices $u, v \in V(G_l)$ for some l, such that $u = (i_1, j_1)$ and $v = (i_2, j_2)$, where $i_1 = i_2 = i'\frac{n}{k}$ for some i' and $j_1 = j'\frac{n}{k}$, $j_2 = (j'+1)\frac{n}{k}$ for some j', or $j_1 = j_2 = j'\frac{n}{k}$ for some j' and $i_1 = i'\frac{n}{k}$, $i_2 = (i'+1)\frac{n}{k}$ for some i', we add an edge between u and v in the set $E(H)$ if and only if there is a path from u to v in G_l and we call such vertices as *corner vertices.*

Before proceeding further, let us introduce a few more notations that will be used later. For $j \in [k]$, let $L_h(i, j)$ denote the set of vertices in $L_h(i)$ in between $L_v(j)$ and $L_v(j+1)$. Similarly we also define $L_v(i, j)$ (see Figure 10.1). For two vertices $x, y \in L_v(i)$, we say $x \prec y$ if x is *below* y in $L_v(i)$. For two vertices $x, y \in L_h(i)$, we say $x \prec y$ if x is *right of* y in $L_h(i)$. Note that we consider these two type of orderings to ensure that for any $x, y \in V(H)$ reachable from s in H, if $x \prec y$, then x will be traversed by our algorithm before y.

Lemma 10.2.1. *There is a path from s to t in G if and only if there is path from s to t in the auxiliary graph H.*

Proof. As every edge (a, b) in H corresponds to a path from a to b in G, so if-part is trivial to see. Now for the only-if-part, consider a path P from s to t in G. P can be decomposed as $P_1 P_2 \cdots P_r$, such that P_i is a path from x_i to x_{i+1}, where x_i is the first vertex on P that belongs to $V(G_l)$ and x_{i+1} be the last vertex on P that also belongs to $V(G_l)$, for some l and in a layered grid graph, for such x_i and x_{i+1}, we have only following two possibilities:

i. x_i and x_{i+1} belong to different horizontal or vertical gridlines, or

ii. x_i and x_{i+1} are two corner vertices.

Now by the construction H, for every i, there must be an edge (x_i, x_{i+1}) in H for both the above cases and hence there is a path from s to t in H as well. □

Now we consider the case when two vertices $x, y \in V(H)$ belong to the same vertical or horizontal gridlines.

Claim 10.2.1. *Let x and y be two vertices contained in either $L_v(i)$ or $L_h(i)$ for some i. Then deciding reachability between x and y in G can be done in log space.*

Proof. Let us consider that $x, y \in L_v(i)$, for some i. As the graph G under consideration is a layered grid graph, if there is a path between x and y, then it must pass through all the vertices in $L_v(i)$ that lies in between x and y. Hence just by exploring the path starting from x through $L_v(i)$, we can check the reachability and it is easy to see that this can be done in log space, because the only thing we need to remember is the current vertex in the path. Same argument will also work when $x, y \in L_h(i)$, for some i and this completes the proof. □

Now we argue on the upper bound of the length of any path in the auxiliary graph H. The idea is to partition the set $V(H)$ into $2k+1$ partitions in such a way that any two consecutive vertices on a path in H lie on two different partitions.

Lemma 10.2.2. *Any path between s and t in H is of length $2k$.*

Proof. Let us first define the sets $D_0, D_1, \cdots, D_{2k}$ (e.g., shaded region in Figure 10.1(b) denotes D_1), where

$$D_l := \{(i,j) | (i'-1)\frac{n}{k} \leq i < i'\frac{n}{k},\ (j'-1)\frac{n}{k} \leq j < j'\frac{n}{k} \text{ and } i'+j' = l+1\}.$$

Now consider $D'_l := D_l \cap V(H)$ for $0 \leq l \leq 2k$. Clearly, $D'_0, D'_1, \cdots, D'_{2k}$ induce a partition on $V(H)$. Now let us take any path $s = x_1 x_2 \cdots x_r = t$, from s to t in H, denoted as P. Observe that by the construction of H, for any two consecutive vertices x_i and x_{i+1} for some i, if $x_i \in D'_l$ for some l, then $x_{i+1} \in D'_{l+1}$ and $s \in D'_0$, $t \in D'_{2k}$. As a consequence, $r = 2k+1$ and hence length of the path P is $2k$. □

10.2.2 Description of the Algorithm

We next give a modified version of DFS that starting at a given vertex, visits the set of vertices reachable from that vertex in the graph H. At every vertex, the traversal visits the set of outgoing edges from that vertex in counter-clockwise order.

In our algorithm we maintain two arrays of size $k+1$ each, say A_v and A_h, one for vertical and the other for horizontal gridlines respectively. For every $i \in [k+1]$, $A_v(i)$ is the *topmost* visited vertex in $L_v(i)$ and analogously $A_h(i)$ is the *leftmost* visited vertex in $L_h(i)$. This choice is guided by the choice of traversal of our algorithm. More precisely, we cycle through the outgoing edges of a vertex in counter-clockwise order.

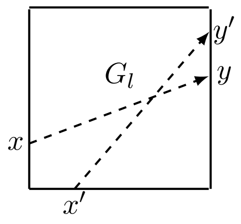


Figure 10.2: Crossing between two paths inside a block G_l

We perform a standard DFS-like procedure, using the tape space to simulate a stack, say S. S keeps track of the path taken to the current vertex from the starting vertex. By Lemma 10.2.2, the maximum length of a path in H is at most $2k$. Whenever we visit a vertex in a vertical gridline (say $L_v(i)$), we check whether the vertex is lower than the i-th entry of A_v. If so, we return to the parent vertex and continue with its next child. Otherwise, we update the i-th entry of A_v to be the current vertex and proceed forward. Similarly when visit a horizontal gridline (say $L_h(i)$), we check whether the current vertex is to the right of the i-th entry of A_h. If so, we return to the parent vertex and continue with its next child. Otherwise, we update the i-th entry of A_h to be the current vertex and proceed. The reason for doing this is to avoid revisiting the subtree rooted at the node of an already visited vertex. The algorithm is formally defined in Algorithm 1.

Lemma 10.2.3. *Let G_l be some block and let x and y be two vertices on the boundary of G_l such that there is a path from x to y in G. Let x' and y' be two other boundary vertices in G_l such that (i) there is a path from x' to y' in G and (ii) x' lies on one segment of the boundary of G_l between vertices x and y and y' lies on the other segment of the boundary. Then there is a path in G from x to y' and from x' to y. Hence, if (x, y) and (x', y') are present in $E(H)$ then so are (x, y') and (x', y).*

Proof. Since G is a layered grid graph hence the paths x to y and x' to y' must lie inside G_l. Also because of planarity, the paths must intersect at some vertex in G_l (See Figure 10.2). Now using this point of intersection, we can easily show the existence of paths from x to y' and from x' to y. □

Lemma 10.2.4 will prove the correctness of Algorithm 1.

Lemma 10.2.4. *Let u and v be two vertices in H. Then starting at u Algorithm 1 visits v if and only if v is reachable from u in H.*

Proof. It is easy to see that every vertex visited by the algorithm is reachable from u since the algorithm proceeds along the edges of H.

By induction on the shortest path length to a vertex, we will show that if a vertex is reachable from u then the algorithm visits that vertex. Let $B_d(u)$ be the set of vertices reachable from u that are at a distance d from u. Assume that the algorithm visits every vertex in $B_{d-1}(u)$. Let x be a vertex in $B_d(u)$. Without loss of generality assume that x is in $L_v(i, j)$ for some i and j. A similar argument can be given if x belongs to a horizontal gridline. Further, let x lie on the right boundary of a block G_l. Let

```
Input  : The auxiliary graph $H$, two vertices $s, t \in V(H)$
Output : YES if there is a path from $s$ to $t$; otherwise NO
Initialize two arrays $A_v$ and $A_h$ and a stack $S$;
Initialize three variables $curr$, $prev$ and $next$ to NULL;
Push $s$ onto $S$;
while $S$ is not empty do
    $curr \leftarrow$ top element of $S$;
    $next \leftarrow$ neighbor of $curr$ next to $prev$ in counter-clockwise order;
    while $next \neq$ NULL do
        /* cycles through neighbors of curr                    */
        if $next = t$ then
            return YES;
        end
        if $next \in L_v(i)$ for some $i$ and $A_v[i] \prec next$ then
            $A_v[i] \leftarrow next$;
            break;
        end
        if $next \in L_h(i)$ for some $i$ and $A_h[i] \prec next$ then
            $A_h[i] \leftarrow next$;
            break;
        end
        $prev \leftarrow next$;
        $next \leftarrow$ neighbor of $curr$ next to $prev$ in counter-clockwise order;
        /* NULL if no more neighbors are present               */
    end
    if $next =$ NULL then
        remove $curr$ from $S$;
        $prev \leftarrow curr$;
    else
        add $next$ to $S$;
        $prev \leftarrow$ NULL;
    end
end
return NO;
```

Algorithm 1: AlgoLGGR: Algorithm for Reachability in the Auxiliary Graph H

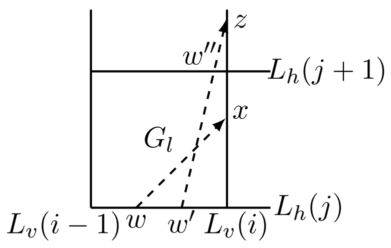


Figure 10.3: Crossing between two paths

$W_x = \{w \in B_{d-1}(u) | (w, x) \in E(H)\}$. Note that by the definition of H, all vertices in W_x lie on the bottom boundary or on the left boundary of G_l.

Suppose the algorithm does not visit x. Since x is reachable from u via a path of length d, therefore W_x is non empty. Let w be the first vertex added to W_x by the algorithm. Then w is either in $L_h(j)$, or in $L_v(i-1)$. Without loss of generality assume w is in $L_h(j)$. Let z be the value in $A_v(i)$ at this stage of the algorithm (that is when w is the current vertex). Since x is not visited hence $x \prec z$. Also this implies that z was visited by the algorithm at an earlier stage of the algorithm. Let w' be the ancestor of z in the DFS tree such that w' is in $L_h(j)$. There must exist such a vertex because z is above the j-th horizontal gridline, that is $L_h(j)$.

Suppose if w' lies to the left of w then by the description of the algorithm, w is visited before w'. Hence x is visited before z. On the other hand, suppose if w' lies to the right of w. Clearly w' cannot lie to the right of vertical gridline $L_v(i)$ since z is reachable from w' and z is in $L_v(i)$. Let w'' be the vertex in $L_h(j+1)$ such that w'' lies in the tree path between w' and z (See Figure 10.3). Observe that all four vertices lie on the boundary of G_l. Now by applying Lemma 10.2.3 to the four vertices w, x, w' and w'' we conclude that there exists a path from w' to x as well. Since $x \prec z$, x must have been visited before z from the vertex w'. In both cases, we see that z cannot be $A_v(i)$ when w is the current vertex. Since z was an arbitrary vertex such that $x \prec z$, the lemma follows. □

The following lemma will help us to achieve a polynomial bound on the running time of Algorithm 1.

Lemma 10.2.5. *Every vertex in the graph H is added to the set S at most once in Algorithm 1.*

Proof. Observe that a vertex u in $L_v(i)$ is added to S only if $A_v(i) \prec u$, and once u is added, $A_v(i)$ is set to u. Also during subsequent stages of the algorithm, if $A_v(i)$ is set to v, then $u \prec v$. Hence $u \prec A_v(i)$. Therefore, u cannot be added to S again.

We give a similar argument if u is in $L_h(i)$. Suppose if u is in $L_v(i)$ for some i and $L_h(j)$ for some j, then we add u only once to S. This check is done in Line 16 of Algorithm 1. However we update both $A_v(i)$ and $A_h(j)$. □

Algorithm 1 does not explicitly compute and store the graph H. Whenever it is queried for an edge (x, y) in H, it recursively runs a reachability query in the corresponding sub grid graph of G such that x is in the bottom left corner and y is in the top right corner of that sub grid graph and produces an answer. The base case is when a query is made to a grid graph of size $k \times k$. For the base case, we run a standard DFS procedure on the $k \times k$ size graph.

In every iteration of the *outer while* loop (Lines 4 – 29) of Algorithm 1, either an element is added or an element is removed from S. Hence by Lemma 10.2.5 the loop iterates at most $4nk$ times. The *inner while* loop (Lines 7 – 21), cycles through all the neighbors of a vertex and hence iterates for at most $2n/k$ times. Each iteration of the inner while loop makes a constant number of calls to check the presence of an edge in a $n/k \times n/k$ sized grid. Let $\mathcal{T}(n)$ and $\mathcal{S}(n)$ be the time and space required to decide reachability in a layered grid graph of size $n \times n$ respectively. Then,

$$\mathcal{T}(n) = \begin{cases} 8n^2(\mathcal{T}(n/k) + O(1)) & \text{if } n > k \\ O(k^2) & \text{otherwise.} \end{cases}$$

Hence, $\mathcal{T}(n) = O\left(n^{3\frac{\log n}{\log k}}\right)$.

Since we do not store any query made to the smaller grids, therefore the space required to check the presence of an edge in H can be reused. A_v and A_h are arrays of size $k+1$ each. By Lemma 10.2.2, the number of elements in S at any stage of the algorithm is bounded by $2k$. Therefore,

$$\mathcal{S}(n) = \begin{cases} \mathcal{S}(n/k) + O(k \log n) & \text{if } n > k \\ O(k^2) & \text{otherwise.} \end{cases}$$

Hence, $\mathcal{S}(n) = O\left(\frac{k}{\log k} \log^2 n + k^2\right)$.

Now given any constant $\epsilon > 0$, if we set $k = n^{\epsilon/2}$, then we get $\mathcal{T}(n) = O(n^{6/\epsilon})$ and $\mathcal{S}(n) = O(n^{\epsilon})$. This proves Theorem 10.5.

Chapter 11

Simultaneous Time-Space Upper Bounds for Certain Problems in Planar Graphs

In this chapter, we study simultaneous time-space bound of several graph theoretic problems that are important from the perspective of complexity theory. The problems include shortest path in directed planar graphs, red-blue path problem in planar DAGs, perfect matching in planar bipartite graphs and some other closely related problems. Let us first recall the main results from Chapter 1 that we are going to prove in this chapter. Following is the result on shortest path problem on directed planar graphs and is restatement of Theorem 7.4.

Theorem 11.1. *For directed planar graphs (containing no negative weight cycle and weights are bounded by polynomial in n) and for any constant $0 < \epsilon < \frac{1}{2}$, there is an algorithm that solves the* ShortestPath *problem in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space, where n is the number of vertices of the given graph.*

Next we discuss a result on designing an algorithm for the RedBluePath problem in planar DAGs. Note that this is restatement of Theorem 7.5.

Theorem 11.2. *For any constant $0 < \epsilon < \frac{1}{2}$, there is a polynomial time algorithm that solves the* RedBluePath *problem for planar DAGs using $O(n^{\frac{1}{2}+\epsilon})$ space.*

Now using the reduction given in [Kul11] and the algorithm stated in the above theorem, we get an algorithm to solve the directed reachability problem for a fairly large class of graphs described in Section 11.2, that takes polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space. Thus we are able to beat the bound given by Barnes, Buss, Ruzzo and Schieber [BBRS92] for such class of graphs.

In this chapter, we also consider EvenPath problem and get the following theorem (restatement of Theorem 7.6) regarding solving EvenPath problem.

Theorem 11.3. *For any constant $0 < \epsilon < \frac{1}{2}$, there is a polynomial time algorithm that solves the* EvenPath *problem in planar DAGs using $O(n^{\frac{1}{2}+\epsilon})$ space.*

Our another contribution is to give an time-space efficient algorithm for perfect matching problem in case of planar bipartite graphs. The following result was stated in chapter 1 as Theorem 7.7.

Theorem 11.4. *In planar bipartite graphs, for any constant* $0 < \epsilon < \frac{1}{2}$*,*

1. PerfectMatching *(Decision + Construction) can be solved in polynomial time and* $O(n^{\frac{1}{2}+\epsilon})$ *space; and*

2. HallObs *(Decision + Construction) can be solved in polynomial time and* $O(n^{\frac{1}{2}+\epsilon})$ *space.*

The main building block of the proof of the above theorem is the Miller and Naor's algorithm [MN95] for perfect matching in planar bipartite graph.

Next we show that the complexity of even perfect matching in planar bipartite graph is same as the perfect matching problem in planar bipartite graph and deciding the presence of odd length cycle in directed planar graph. Thus we get the following theorem for EvenPM problem.

Theorem 11.5. *For any constant* $0 < \epsilon < \frac{1}{2}$*, there exists an algorithm that solves the* EvenPM *problem for planar bipartite graphs in polynomial time and* $O(n^{\frac{1}{2}+\epsilon})$ *space.*

We want to remind the reader that the above theorem is a restatement of Theorem 7.8.

11.1 Shortest Path Problem in Directed Planar Graphs

Let ShortestPath be the problem of computing the shortest distance and the corresponding path between a pair vertices in a graph. For a given graph $G = (V, E)$ with a weight function $w : E \to \mathbb{R}$ (negative weights are also allowed), and two vertices s and t, let $dist_G^w(s,t)$ denote the shortest distance and $path_G^w(s,t)$ denote the shortest path from s to t. We will consider the weight assigned to an edge is bounded by some polynomial in n, where n is the number of vertices in G. Note that single-source shortest path problem and all-pair shortest path problem can be solved by executing the algorithm used for the ShortestPath problem polynomially many times. Now we consider the ShortestPath problem when the given graph G is directed planar.

Proof of Theorem 11.1. Let $G = (V, E)$ be the given directed planar graph, where $|V| = n$. Consider any constant $0 < \epsilon < 1/2$. Let $\overline{S}$ be a $(n^{\frac{1}{2}+\epsilon}, n^{-\epsilon})$-separator computed by the procedure `PlanarSeparatorFamily` on the underlying undirected graph of G and $S = \overline{S} \cup \{s, t\}$. Let us define an array C_s of size $|S|$, where $C_s[i]$ will store the distance of i-th vertex, denoted by v_i, of the set S from the vertex s. Initially $C_s[s] = 0$, and for all $v_i \in S$ such that $v_i \neq s$, $C_s[i] = \infty$. For calculating the shortest distance, we use `Bellman-Ford` algorithm if the graph is small; otherwise, we calculate the shortest distance of the vertices in S from s through each connected component of $G[V \setminus S]$ and then use those distances to calculate the shortest distance from s to t in the overall graph. We do this recursively to achieve the desired time and space bound. To find the shortest distance between s and t in the given graph G, we run the procedure `PlanarDist` (Algorithm 2) with the input (G, s, t, n, S, C_s), where n is the number of vertices of G.

```
   Input    : $G' = (V', E'), s', t', n, T, A$
   Output : $dist^{w}_{G'}(s', t')$
   /* Let $r' = n'^{(-\epsilon)}$, $|V'| = n'$.                                    */
1  if $n' \leq n^{\frac{1}{2}}$ then
2      Run Bellman-Ford($G', s', t'$) using the values stored in $A$ and return
       $dist^{w}_{G'}(s', t')$;
3  else
4      Run PlanarSeparatorFamily on the underlying undirected graph of $G'$ to
        compute a $(n'^{(\frac{1}{2}+\epsilon)}, r')$-separator and let us denote this separator by $\overline{S'}$;
5      Set $S' := \overline{S'} \cup \{s', t'\}$;
6      Define an array $C_{s'}$ of size $|S'|$. $C_{s'}[i]$ will store $dist^{w}_{G'}(s', v_i)$, where $v_i$ is the $i$-th
        vertex of the set $S'$ and set $\forall_{v_i \in T}, C_{s'}[i] = A[i], \forall_{v_i \notin T}, C_{s'}[i] = \infty$
7      for $round = 1$ to $|S'|$ do
8          for every $x \in V'$ do
9              for every $v \in S'$ do
                   /* Let $V_x$ be the set of vertices of the undirected
                      version of $G[V' \setminus S']$'s connected component containing
                      the vertex $x$.                                          */
10                 Run PlanarDist($G[V_x \cup S'], s', v, n, S', C_{s'}$);
11                 Update $C_{s'}$;
12             end
13         end
14     end
15 end
```

Algorithm 2: Algorithm `PlanarDist`: Shortest Distance in Directed Planar Graphs

We use the procedure `PlanarDist` (Algorithm 2) as a subroutine to report the shortest path. The algorithm is stated as `PlanarShortPath` (Algorithm 3) and to report the shortest path between s and t in G, we run this algorithm with the input (G, s, t, n, S, C_s).

```
   Input  : $G' = (V', E'), s', t', n, T, A$
   Output : $path^w_{G'}(s', t')$ in reverse order
 1 Run PlanarDist($G', s', t', n, T, A$);
   /* Let $\overline{S'}$ be the corresponding separator obtained by running the
      procedure PlanarSeparatorFamily and $S' = \overline{S'} \cup \{s', t'\}$. $C_{s'}$ be the
      corresponding array storing the shortest distances of vertices in
      the separator from $s'$ and $N(t')$ be the set of neighbor vertices of
      the vertex $t'$.                                                    */
 2 Define an array $C_{t'}$ of size $|S'|$. Initialize $C_{t'}[t'] = 0$ and $\forall_{v_i \neq s'}, C_s[i] = \infty$.
 3 for every $x \in N(t')$ do
 4     for every $v \in \overline{S'}$ do
           /* Let $G_{rev}$ be the graph with the same set of vertices as $G$ but
              the direction of the edges are reversed.                    */
 5         Run PlanarDist($G_{rev}[V_x \cup S'], t', v, n, S', C_{t'}$);
 6     end
 7 end
   /* Let $v' \in \overline{S'}$ such that $dist^w_{G_{rev}[V_{x'} \cup S']}(t', v') + dist^w_{G'}(s', v') = dist^w_{G'}(s', t')$.
      */
 8 if $|V_{x'} \cup S'| \leq n^{\frac{1}{2}}$ then
 9     Run Bellman-Ford($G_{rev}[V_{x'} \cup S'], t', v'$) and report $path^w_{G_{rev}[V_{x'} \cup S']}(t', v')$;
10 else
11     Run PlanarShortPath($G_{rev}[V_{x'} \cup S'], t', v', n, S', C_{t'}$);
12 end
13 Reinitialize $A$ and Run PlanarShortPath($G', s', v', n, T, A$);
```

Algorithm 3: Algorithm `PlanarShortPath`: Report Shortest Path in Directed Planar Graphs

In Algorithm 2, within the loop (Lines 8 – 13), we evaluate the shortest distance of all the vertices $v_i \in S$ from s through each connected component of $G[V \setminus S]$. By update C_s (Line 11), we mean that update the entry in $C_s[i]$ for some i, if the currently calculated distance of v_i from s is smaller than the previously stored one. We run the loop (Lines 8 – 13) total $|S|$ number of times and the reason for that is mentioned while proving the correctness of the algorithm.

Let $\mathcal{S}(n)$ and $\mathcal{T}(n)$ denote the space and time complexity functions for computing shortest distance in a graphs containing n vertices. Since $(1-\epsilon)^k \leq \frac{1}{2}$ for $k = O(\frac{1}{\epsilon})$, the depth of the recursion is $O(\frac{1}{\epsilon})$. Also, $|V_x \cup S'| \leq 2n'^{(1-\epsilon)}$. This gives us the following recurrence relation:

$$\mathcal{S}(n') = \begin{cases} \tilde{\mathrm{O}}(n'^{(\frac{1}{2}+\epsilon)}) + \mathcal{S}(2n'^{(1-\epsilon)}) & \text{if } n' > n^{\frac{1}{2}} \\ \tilde{\mathrm{O}}(n^{\frac{1}{2}}) & \text{otherwise} \end{cases}$$

Thus, $\mathcal{S}(n) = O(\frac{1}{\epsilon})\tilde{\mathrm{O}}(n^{\frac{1}{2}+\epsilon}) = \tilde{\mathrm{O}}(n^{\frac{1}{2}+\epsilon})$.
For time analysis, we get the following recurrence relation:

$$\mathcal{T}(n') = \begin{cases} q(n)(p_1(n')\mathcal{T}(2n'^{(1-\epsilon)}) + p_2(n')) & \text{if } n' > n^{\frac{1}{2}} \\ q(n)\tilde{\mathrm{O}}(n^{\frac{1}{2}}) & \text{otherwise} \end{cases}$$

where $q(n)$, $p_1(n)$ and $p_2(n)$ are some polynomials in n. As the recursion depth is bounded by $O(\frac{1}{\epsilon})$ (a constant) and the subroutine `PlanarDist` (Algorithm 2) is called polynomial many times, we have $\mathcal{T}(n) = p(n)$ for some polynomial $p(n)$. Using a similar analysis, it can easily be seen that the algorithm for reporting the shortest path also uses $\widetilde{O}(n^{\frac{1}{2}+\epsilon})$ space and polynomial time.

Proof of correctness: Next we show the correctness of the above algorithms using induction. Let $G' = (V', E'), s', t', n, S, C_s$ be an instance of the procedure `PlanarDist`. When $n' \leq n^{\frac{1}{2}}$, the correct answer is given since it is just the execution of the `Bellman-Ford` algorithm. Now consider the shortest path P from s to t, which can be decomposed as $s = v_0 \xrightarrow{P_1} v_1 \xrightarrow{P_2} v_2 \cdots v_k \xrightarrow{P_k} v_{k+1} = t$, where each P_i, for $1 \leq i \leq k$, is the shortest path between v_{i-1} and v_i through a $O(n'^{(1-\epsilon)})$ sized connected region. By induction on n, we can say that $s = v_0 \xrightarrow{P_1} v_1 \xrightarrow{P_2} v_2 \cdots v_{i-1} \xrightarrow{P_i} v_i$ is the shortest path from s to v_i. As the size of the separator under consideration is $\widetilde{O}(n^{\frac{1}{2}+\epsilon})$ and each path going from one $O(n'^{(1-\epsilon)})$ sized region to other must pass through a vertex in the separator $\overline{S}$, so $k \leq |\overline{S}|$. Thus the execution of loop [Lines 8 – 13] in Algorithm 2 total $|S|$ number of times suffices to output the shortest distance between s and t. The above argument can easily be proved by inducting on path length and is same as the proof of correctness of standard `Bellman-Ford` algorithm [KT05].

Using a similar argument, it is easy to see that the procedure `PlanarShortPath` (Algorithm 3) will correctly output the shortest path from s to t in the reverse order. □

11.1.1 Detecting Negative Weight Cycle in Directed Planar Graphs

If we determine the shortest path from s to all other vertices then negative cycle must lie in any one of these paths and shortest distance of that path is negative infinity. Using this fact we can also detect negative weight cycle in a given graph. Now if there exists a negative weight cycle in the given graph, then either that will be completely inside $O(n^{(1-\epsilon)})$ sized region or it must pass through at least two vertices of the separator family. To detect the negative weight cycle we use a slightly modification of the procedure `PlanarDist` (Algorithm 2). In the modified version we run the procedure `PlanarDist` (Algorithm 2) slightly more than $|S|$ times and if in the last run any v, the value of $C_s[v]$ changes then we can infer that there is a negative weight cycle and which lies in the path of s to v. We can report the negative weight cycle just using the procedure `PlanarShortPath` (Algorithm 3). Thus the detecting and reporting negative weight cycle problem also have the same space and time complexity (up to polynomial blow up) as that of shortest path problem.

Corollary 11.1.1. *For directed planar graphs, for any constant $0 < \epsilon < 1/2$, there is an algorithm that solves the problem of detecting negative weight cycle in polynomial time and uses $O(n^{\frac{1}{2}+\epsilon})$ space, where n is the number of vertices of the given graph G.*

11.2 Red-Blue Path Problem

11.2.1 Deciding Red-Blue Path in Planar DAGs

Given a directed graph G with each edge colored either red or blue and two vertices s and t, a *red-blue path* denotes a path that alternates between red and blue edges and the RedBluePath problem decides whether there exists a directed red-blue path from s to t such that the first edge is red and the last edge is blue. The RedBluePath problem is a generalization of the reachability problem in graphs, however this problem is NL-complete even when restricted to planar DAGs [Kul11]. This makes it an interesting problem in the area of space bounded complexity as to the best of our knowledge, this is the only "reachability-like" problem in planar graphs that is hard for NL. Before going into the proof of Theorem 11.2, let us recall one notation that given a directed graph G, we denote the underlying undirected graph by $\hat{G}$. We use this notation frequently in this section.

Proof of Theorem 11.2. Consider a planar DAG G. Let $\overline{S}$ be a $(n^{\frac{1}{2}+\epsilon}, n^{-\epsilon})$-separator computed by `PlanarSeparatorFamily` on $\hat{G}$ and let $S = \overline{S} \cup \{s, t\}$. For the sake of convenience, we associate two numerical values to the edge colors – 0 to red and 1 to blue. We run the subroutine `RedBluePathDetect` (Algorithm 6) with the input $(G, s, t, n, 0, 1)$ and if the returned value is true, then we say that there is a directed red-blue path from s to t such that the first edge is red and last one is blue; otherwise we say that there is no such path. In Algorithm 5, we use the notation $(u, v) \in^{(init,temp)} \overline{E'}$ to decide whether there is a red-blue path from u to v that starts with an edge of color value *init* and ends with an edge of color value *temp*.

```
  Input    : $G' = (V', E'), s', t', init, final$
  Output : True if there is a red-blue path from $s'$ and $t'$ starts with $init$ and ends
               with $final$
  /* Use two sets- $N_i$, for $i = 0, 1$, to store all the vertices that have
     been explored with the color value $i$                                   */
1 if $s' \notin N_{init}$ then
2    Add $s'$ in $N_{init}$;
3    for each edge $(s', v) \in E'$ of color value $init$ do
4       if $v = t'$ and $init = final$ then
5          Return true;
6       end
7       Run ColoredDFS($G', v, t', init + 1 (mod\ 2), final$);
8    end
9 end
```

Algorithm 4: Algorithm `ColoredDFS`: One of the Building Blocks of `RedBluePathDetect`

In Algorithm 4, we use general DFS type search to check the presence of a red-blue path between any two given vertices s' and t'. The only difference with DFS search is that here we explore edges such that color of the edges alternates between red and blue. If we start from a vertex s', then the for loop (Lines 3 – 8) explore the path starting from s' such that first edge of the path is of specified color. In the main algorithm (Algorithm 6),

```
Input  : $\overline{G'} = (\overline{V}', \overline{E'}), G', s', t', init, final$
Output : True if there is a red-blue path from $s'$ and $t'$ starts with $init$ and ends
         with $final$
/* Use two sets- $R_i$, for $i = 0, 1$, to store all the vertices that have
   been explored with the color value $i$                                   */
1  if $s' \notin R_{init}$ then
2      Add $s'$ in $R_{init}$;
3      for each $(s', v) \in^{(init,temp)} \overline{E'}$ for each $temp \in \{0, 1\}$ do
4          if $v = t'$ and $temp = final$ then
5              Return true;
6          end
7          Run ModifiedColoredDFS($\overline{G}', G', v, t', temp + 1 \pmod 2, final$);
8      end
9  end
/* ``$(u, v) \in^{(init,temp)} \overline{E'}$?'' query will be solved using the following
   procedure                                                                */
10 for every $a \in V$ do
       /* $V$ be the set of vertices of $G'$                                */
       /* $V_a$ = the set of vertices of $\hat{H}$'s connected component containing
          $a$, where $H = G[V \setminus \overline{V}']$                          */
11     if RedBluePathDetect($G[V_a \cup \overline{V}'], u, v, n, init, temp$) is true then
12         Return true for the query;
13     end
14 end
15 Return false for the query;
/* End of the query procedure                                               */
```

Algorithm 5: Algorithm `ModifiedColoredDFS`: One of the Building Blocks of `RedBluePathDetect`

```
Input  : $G', s', t', n, init, final$
Output : True if there is a red-blue path from $s'$ and $t'$ starts with $init$ and ends
         with $final$
1  if $n' \leq n^{\frac{1}{2}}$ then
2      Run ColoredDFS($G', s', t', init, final$);
3  else
       /* let $r' = n'^{(-\epsilon)}$                                          */
4      Run PlanarSeparatorFamily on $\hat{G}'$ to compute a $(n'^{(\frac{1}{2}+\epsilon)}, r')$-separator and let
       us denote this separator by $\overline{S'}$;
5      Run ModifiedColoredDFS($\overline{G'} = (\overline{S'} \cup \{s', t'\}, \overline{E'}), G', s', t', init, final$);
6  end
```

Algorithm 6: Algorithm `RedBluePathDetect`: Algorithm for Red-Blue Path in planar DAG

we use the procedure `ColoredDFS` (Algorithm 4) as a base case, i.e., when the input graph is small in size (is of size $n^{1/2}$). Otherwise, we first compute S and then run Algorithm 5 on the auxiliary graph $\overline{G} = (S, \overline{E})$. Algorithm 6 does not explicitly store the graph $\overline{G}$. Whenever it is queried with a pair of vertices to check the presence of an edge within the graph $\overline{G}$, it recursively runs Algorithm 6 on all the connected components of $G[V \setminus S]$ separately (Lines 10 – 15 of Algorithm 5) and produces an answer. Finally, we perform same DFS like search as in Algorithm 4 on $\overline{G}$ (Lines 1 – 9 of Algorithm 5).

In the base case, we use Algorithm 4 which takes linear space and polynomial time. Thus due to the restriction of the size of the graph in the base case, we have $\widetilde{O}(n^{1/2})$ space and polynomial time complexity. The sets N_0 and N_1 of the procedure `ColoredDFS` (Algorithm 4) only store all the vertices of the input graph and we run the procedure `ColoredDFS` (Algorithm 4) on a graph with $n^{1/2}$ vertices and it visits all the edges of the input graph at most once which results in the polynomial time requirement.

Let $\mathcal{S}(n)$ and $\mathcal{T}(n)$ denote its space and time complexity functions for input graphs containing n vertices. Since $(1-\epsilon)^k \leq \frac{1}{2}$ for $k = O(\frac{1}{\epsilon})$, the depth of the recursion is $O(\frac{1}{\epsilon})$. Also, $|V_a \cup \overline{S'}| \leq 2n'^{(1-\epsilon)}$. This gives us the following recurrence relation:

$$\mathcal{S}(n') = \begin{cases} \tilde{O}(n'^{(\frac{1}{2}+\epsilon)}) + \mathcal{S}(2n'^{(1-\epsilon)}) & \text{if } n' > n^{\frac{1}{2}} \\ \tilde{O}(n^{\frac{1}{2}}) & \text{otherwise} \end{cases}$$

Thus, $\mathcal{S}(n) = O(\frac{1}{\epsilon})\tilde{O}(n^{\frac{1}{2}+\epsilon}) = \tilde{O}(n^{\frac{1}{2}+\epsilon})$.
For time analysis, we get the following recurrence relation:

$$\mathcal{T}(n') = \begin{cases} q(n)(p_1(n')\mathcal{T}(2n'^{(1-\epsilon)}) + p_2(n')) & \text{if } n' > n^{\frac{1}{2}} \\ q(n)\tilde{O}(n^{\frac{1}{2}}) & \text{otherwise} \end{cases}$$

where $q(n)$, $p_1(n)$ and $p_2(n)$ are some polynomials in n. As the recursion depth is bounded by $O(\frac{1}{\epsilon})$ (a constant), we have $\mathcal{T}(n) = p(n)^{O(\frac{1}{\epsilon})}$ for some polynomial $p(n)$.

Proof of correctness: We now give a brief idea about the correctness of this algorithm. In the base case, we use technique similar to DFS just by alternatively exploring red and blue edges and thus this process gives us a path where two consecutive edges are of different colors. Otherwise, we also do a DFS like search by alternatively viewing red and blue edges and we do this search on the graph $H = (\overline{S'} \cup \{s,t\}, \overline{E'})$. By this process, we decide on presence of a path in H from s to t such that two consecutive edges are of different colors in G and the edge coming out from s is red and the edge going in at t is blue. This is enough as each path P in G must be broken down into the parts $P_1, P_2, \cdots, P_k$ and each P_i must be a sequence of edges that starts and ends at some vertices of $\overline{S'} \cup \{s,t\}$ and also alternates in color. We find each such P_i, just by considering each connected component of $G(V' \setminus \overline{S'})$ and repeating the same steps recursively. □

Due to [Kul11], we know that the reachability problem in directed graphs reduces to RedBluePath in planar DAGs. For the class of graphs in which this reduction results the sub-quadratic increase in the number of vertices, we have an algorithm for reachability problem that takes sublinear space and polynomial time. As a special case of this we can state the following theorem.

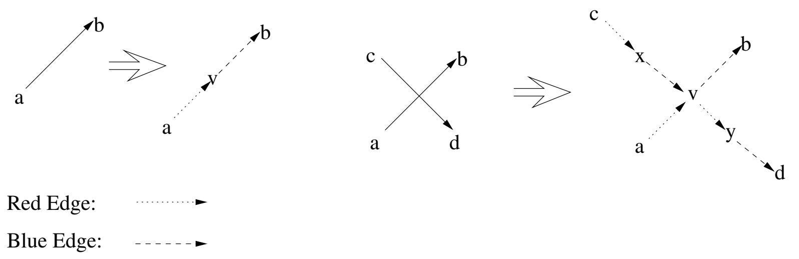


Figure 11.1: Red-Blue Edge Gadget

Theorem 11.6. *Given a directed acyclic graph $G = (V, E)$, where $|E| = \widetilde{O}(n)$, with a drawing in a plane such that the number of edge crossings is $\widetilde{O}(n)$ and two vertices s and t, then for any constant $0 < \epsilon < \frac{1}{2}$, there is an algorithm that decides whether there is a path from s to t or not. This algorithm runs in polynomial time and uses $O(n^{\frac{1}{2}+\epsilon})$ space, where n is the number of vertices of G.*

Proof. We consider a reduction similar to the reduction from directed reachability problem to RedBluePath problem in planar DAG given in [Kul11]. We do the following:

i. insert new vertices in between edges of G so that in the resulting graph each edge takes part in only one crossing, and

ii. replace each crossing of the resulting graph with a *planarizing gadget* as in Figure 11.1 and also replace each edge without any crossing with two edges as shown in Figure 11.1.

Denote the resulting graph as G_{planar} and the corresponding vertices of s and t as s' and t'. It is easy to see that there is a bijection between $s - t$ paths in G and $s' - t'$ red-blue paths in G_{planar} that starting with a red edge and ending with a blue edge.

If the drawing of the given graph G contains k edge crossings, then step (i) will introduce at most $2k$ many new vertices and say after this step the number of edges becomes m. Then step (ii) will introduce at most $(2m + 3k)$ many vertices. It is clear from the reduction itself that $m = \widetilde{O}(n)$ and thus the graph G_{planar} contains $\widetilde{O}(n)$ many vertices. Now by applying the procedure `RedBluePathDetect` (Algorithm 6) on G_{planar}, we get the desired result.

□

A large class of graphs will satisfy the conditions specified in Theorem 11.6. We now explicitly give an example of one such class of graphs. Before that, we give some definitions. *Crossing number* of a graph G, denoted as $cr(G)$, is the lowest number of edge crossings (or the crossing point of two edges) of a drawing of the graph G in a plane. A graph is said to be *k-planar* if it can be drawn on the plane in such a way that each edge has at most k crossing point (where it crosses a single edge). It is known from [PT97] that a k-planar graph with n vertices has at most $O(n\sqrt{k})$ many edges. Note that a k-planar graph has crossing number at most mk, where m is the number of edges. Now we can state the following corollary.

Corollary 11.2.1. *Given a directed acyclic graph, which is k-planar, where $k = O(\log^c n)$, for some constant c, with a drawing in a plane having minimum number of edge crossings* *and two vertices s and t, then for any constant $0 < \epsilon < \frac{1}{2}$, there is an algorithm that decides whether there is a path from s to t or not. This algorithm runs in polynomial time and uses $O(n^{\frac{1}{2}+\epsilon})$ space, where n is the number of vertices of the given graph.*

11.2.2 Deciding Even Path in Planar DAGs

Given directed graph G and two vertices s and t, EvenPath is the problem of deciding the presence of a (simple) directed path from s to t, that contains even number of edges. We can view this problem as a relaxation of RedBluePath problem, because a path starting with red edge and ending with blue edge is always of even length. In this section, we establish a relation between EvenPath problem in planar DAG with detecting a odd length cycle in a directed planar graph with weight one (can also be viewed as an unweighted graph).

Lemma 11.2.1. *For directed planar graphs, for any constant $0 < \epsilon < \frac{1}{2}$, there is an algorithm that solves the problem of deciding the presence of odd length cycle in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space, where n is the number of vertices of the given graph.*

The above lemma is true due to the fact that we can do BFS efficiently for undirected planar graph and it is enough to detect odd length cycle in each of the strong components of the undirected version of the given directed planar graph. For undirected graph, presence of odd length cycle can be detected using BFS algorithm and then put red and blue colors on the vertices such that vertices in the consecutive levels get the opposite colors. After coloring of vertices if there exists a monochromatic edge (edge where both vertices get the same color), then we can conclude that there is an odd length cycle in the graph otherwise there is no odd length cycle. But this is not the case for general directed graphs because if we use the same approach we might end up with finding a set of edges which form an cycle only if we ignore directions. However, the following proposition will help us to detect odd length cycle in directed graph.

In the following proposition, we use $u \to v$ to denote a directed edge (u, v) and $x \xrightarrow{P} y$ to denote a directed path P from a vertex x to y.

Proposition 11.2.1. *A strongly connected directed graph contains an odd length cycle if and only if the underlying undirected graph contains an odd length cycle.*

Proof. The forward direction follows trivially. Now to prove the converse direction, we will use the induction arguments on the length of the odd cycle in the undirected version of the graph. The base case is when the undirected version of the graph contains a 3-length cycle. If the undirected edges present in the undirected cycle also form directed cycle when we consider the corresponding edges in the directed graph, then there is nothing to prove. But if this is not the case, then the Figure 11.2 will depict the possible scenarios. As the graph is strongly connected, so there must be a path P from t to s and if this path does not pass through the vertex x, then any one of the following two cycles $s \to t \xrightarrow{P} s$ or $s \to x \to t \xrightarrow{P} s$ must be of odd length. Now suppose P contains the vertex x and thus $P = P_1P_2$, where P_1 is the path from t to x and P_2 is the path from x to s. It is easy to see that all the three cycles $s \to t \xrightarrow{P} s$, $x \to t \xrightarrow{P_1} x$ and $s \to x \xrightarrow{P_2} s$ cannot be of even length.

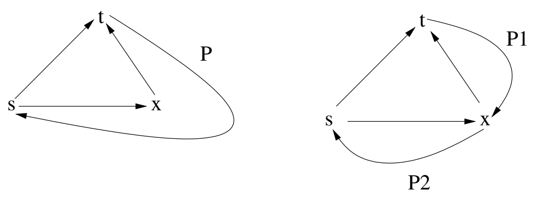


Figure 11.2: For undirected cycle of length 3

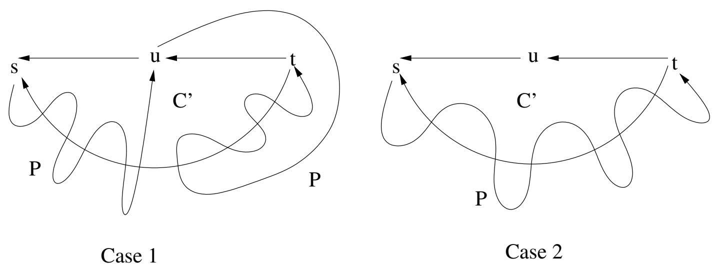


Figure 11.3: For undirected cycle of length $(k+2)$

Now by induction hypothesis, assume that if the undirected version has a cycle of k-length (k odd), then there exists an odd length cycle in the original directed graph.

Now let us prove this induction hypothesis for any undirected cycle of length $(k+2)$. Consider the corresponding edges in the directed graph and without loss of generality assume that this is not a directed cycle. As $(k+2)$ is odd, so there must be one position at which two consecutive edges are in the same direction. Now contract these two edges in both directed and undirected version of the graph and consider the resulting k-length cycle in the undirected graph. So according to the induction hypothesis, there must be one odd length cycle C in the resulting directed graph. Now if C does not contain the vertex u (where we contract the two edges), then expanding the contracted edges will not destroy that cycle and we get our desired odd length cycle in the directed version of the graph. But if this is not the case, then consider C after expanding those two contracted edges($t \rightarrow u \rightarrow s$), say the resulting portion is C'. If C' is a cycle, then there is nothing more to do. But if not, then consider the path P from s to t (there must be such path as the graph is strongly connected). Now there will be two possible cases: either P contains u or not. It is easy to see that for both the possible cases (case 1 and case 2 of Figure 11.3 and in that figure every crossing of two paths denotes a vertex), all cycles generated by C' and P cannot be of even length. In case 1, if all the cycles generated by the paths $s \xrightarrow{P} u$ and $t \xrightarrow{C'} s$ and all the cycles generated by the paths $u \xrightarrow{P} t$ and $t \xrightarrow{C'} s$ are of even length, then as $t \xrightarrow{C'} s$ is of odd length, so the path $s \xrightarrow{P} u \xrightarrow{P} t$ must be of odd length. And then one of the following two cycles $s \xrightarrow{P} u \rightarrow s$ and $u \xrightarrow{P} t \rightarrow u$ is of odd length. Similarly in case 2, if all the cycles generated by $s \xrightarrow{P} t$ and $t \xrightarrow{C'} s$ are of odd length, then the path $s \xrightarrow{P} t$ is of odd length and so the cycle $s \xrightarrow{P} t \rightarrow u \rightarrow s$ is of odd length. □

Proof of Proposition 11.2.1. In a directed planar graph, any cycle cannot be part of two different strong component, so checking presence of odd cycle is same as checking presence of odd cycle in each of its strong components. Constructing strong components of a directed planar graph can be done by polynomial many times execution of `PlanarReach` algorithm (See Theorem 8.3), because a strong component will contain vertices x, y if and only if `PlanarReach`(G, x, y, n) and `PlanarReach`(G, y, x, n) both return "yes". And thus strong component construction step will take $\widetilde{O}(n^{\frac{1}{2}+\epsilon})$ space and polynomial time. After constructing strong components, it is enough to check presence of odd cycle in the underlying undirected graph (according to Proposition 11.2.1). So now on, without loss of generality, we can assume that the given graph G is strongly connected. Now execute `UPlanarOddCycle`$(\hat{G}, s, n)$ (Algorithm 7) after setting the color of s (any arbitrary vertex) to red. Here we adopt the well known technique used to find the presence of odd length cycle in a graph using BFS together with coloring of vertices. In Algorithm 7, instead of storing color values for all the vertices, we only stores color values for the vertices present in the separator (Line 11) and we do the coloring recursively by considering the smaller connected components (Line 10). The algorithm is formally defined in Algorithm 7.

```
   Input  : $G' = (V', E'), s', n$, where $G'$ is an undirected graph
   Output : "Yes" if there is an odd length cycle
 1 if $n' \leq n^{\frac{1}{2}}$ then
 2     Run BFS($G'$ , $s'$) and color the vertices with red and blue such that vertices in
         the alternate layer get the different color starting with a vertex that is already
         colored;
 3     if there is a conflict between stored color of a vertex and the new color of that
         vertex or there is an edge between same colored vertices then
 4         return "yes";
 5     end
 6 else
       /* let $r' = n'^{(-\epsilon)}$                                              */
 7     Run PlanarSeparatorFamily on $\hat{G}'$ to compute a $(n'^{(\frac{1}{2}+\epsilon)}, r')$-separator and let
         us denote this separator by $\overline{S'}$;
 8     Set $S' := \overline{S'} \cup \{s'\}$;
 9     for every $x \in V'$ do
           /* $V_x$ = the set of vertices of $\hat{H}$'s connected component
              containing $x$, where $H = G[V' \setminus S']$                    */
10         Run UPlanarOddCycle($G[V_x \cup S'], s', n$);
11         Store color of the vertices of $S'$ in an array of size $|S'|$;
12     end
13 end
```

Algorithm 7: Algorithm `OddCycleUndirectedPlanar`: Checking Presence of Odd Cycle in an Undirected Planar Graph

By doing the similar type of analysis as that of `RedBluePathDetect` (Algorithm 6), it can be shown that the procedure `UPlanarOddCycle` (Algorithm 7) takes $O(n^{\frac{1}{2}+\epsilon})$ space and polynomial time when input graph has n vertices. So over all, space complexity of detecting odd length cycle in directed planar graph is $O(n^{\frac{1}{2}+\epsilon})$ and time complexity is polynomial in n.

Now we argue on the correctness of the procedure `UPlanarOddCycle` (Algorithm 7).

This algorithm will return “yes” in two cases. First case is when there is an odd length cycle completely inside a small region ($n' \leq n^{\frac{1}{2}}$) and so there is nothing to prove for this case as it is an well known application of BFS algorithm [KT05]. Now in the second case, a vertex v in the separator family will get two conflicting colors means that there exists at least one vertex u in the separator family such that there are two vertex disjoint odd as well as even length path from u to v and as a result, both of these paths together will form an odd length cycle. □

Now we are ready to prove the main theorem of this subsection.

of Theorem 11.3. Given a planar DAG G and two vertices s and t, first report a path from s to t, say P, which can easily be done by polynomially many invocation of the algorithm `PlanarReach` of Theorem 8.3 and thus requires polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space. If the path P is not of even length, then construct a directed graph G' which has the same vertices and edges as G except the edges in path P, instead we do the following: if there is an edge (u, v) in P, then we add an edge (v, u) in G'. Now we can observe that the new graph G' is a directed planar graph.

Claim 11.2.1. *G has an even length path if and only if G' has an odd length cycle.*

Proof. Suppose G' has an odd length cycle, then that cycle must contains the reverse edges of P in G as the graph G under consideration is a planar DAG and thus does not contain any directed cycle. Denote the reverse of the path P by P_{rev}. Now let us assume that the odd cycle C' contains a portion of P_{rev} (See Figure 11.4). Assume that the cycle C' enters into P_{rev} at x (can be t) and leaves P_{rev} at y (can be s). Then in the original graph G, the path $s \xrightarrow{P} y \xrightarrow{C'} x \xrightarrow{P} t$ is of even length. Now for the converse, let us

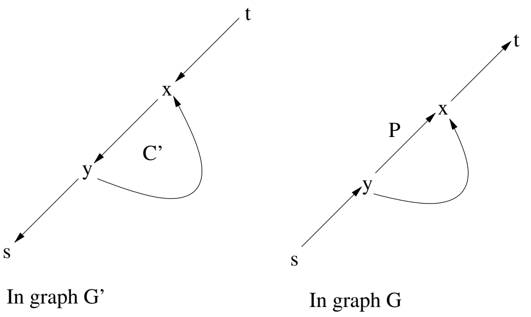


Figure 11.4: When G' contains an odd length cycle

assume that there exists an even length path P_1 from s to t in G. Both the paths P and P_1 may or may not share some edges and without loss of generality we can assume that they share some edges (See Figure 11.5). Now if we consider all the cycles formed by P_{rev} and portions of P_1 in G', then it is easy to see that all the cycles cannot be of even length until length of P and P_1 both are of same parity (either both odd or both even), but this is not the case. □

Now we can check the presence of an odd length cycle in the graph G' in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space (by Lemma 11.2.1). □

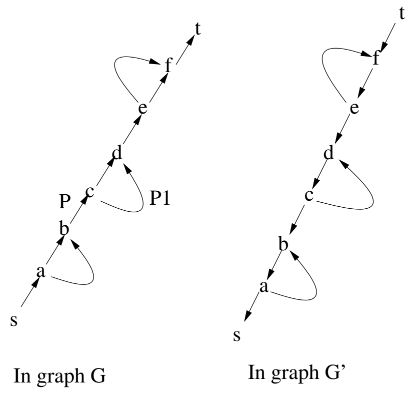


Figure 11.5: When G contains an even length $s - t$ path

11.3 Perfect Matching in Planar Bipartite Graphs

11.3.1 Finding a Perfect Matching

In a graph G, a *matching* is a set of vertex disjoint edges and the end-points of these edges are said to be *matched.* A *perfect matching* is a matching where every vertex of the graph is matched. In this section, we consider the following two matching problems.

1. PerfectMatching (Decision): given a graph G, decide whether G contains a perfect matching, and

2. PerfectMatching (Construction): given a graph G, construct a perfect matching (if exists).

We first discuss the Miller and Naor's algorithms (Algorithm 8 and 9) for solving the decision version of the perfect matching problem in planar bipartite (undirected) graphs and also for constructing a perfect matching (Algorithm 10) [MN95]. Our main observation is that all of the above three algorithms can be implemented in polynomial time and using $O(n^{\frac{1}{2}+\epsilon})$ space.

Before discussing the algorithms, we first define a few terminologies which will be used later.

Definition 11.1 (Capacity-Demand Graph)**.** *A* capacity-demand graph *of an undirected graph $G = (V, E)$ is defined as a triple $(G' = (V', E'), c, d)$, where*

$$E' = \{(u, v) \mid \{u, v\} \in E\}$$

and every edge $(u, v) \in E'$ is assigned a real valued capacity $c(u, v)$*, and every vertex $v \in V'$ is assigned a real valued* demand $d(v)$*.*

Note that a capacity-demand graph of an undirected graph is a directed graph.

Definition 11.2 (Pseudo-flow in a capacity-demand graph)**.** *A* pseudo-flow *in a capacity-demand graph $(G = (V, E), c, d)$ is defined as a function $f : E \to \mathbb{R}$ such that the following holds:*

i. for every edge $(u,v) \in E$*,* $f(u,v) = -f(v,u)$*, and*

ii. for every vertex $v \in V$*,* $\sum\limits_{w \in V:(v,w) \in E} f(v,w) = d(v)$*.*

Definition 11.3 (Flow in a capacity-demand graph)**.** *A* flow *in a capacity-demand graph* $(G = (V,E), c, d)$ *is defined as a function* $f : E \to \mathbb{R}$ *such that:*

i. f *is a pseudo-flow in* (G,c,d)*, and*

ii. for every $(u,v) \in E$*,* $f(u,v) \leq c(u,v)$*.*

A *zero-demand* graph (G,c) is a capacity-demand graph where $d(v) = 0$, for all the vertices $v \in V$.

The *dual graph* of a planar graph $G = (V,E)$ is a graph that has a vertex corresponding to each face of G and an edge corresponding to each edge of $e \in E$. The edge corresponding to $e \in E$ in the dual graph connects the vertices corresponding to the two faces of G which have e as common on their boundaries.

Definition 11.4 (Directed Dual)**.** *Suppose the dual of a undirected planar graph* $G = (V,E)$ *with respect to a fixed embedding is denoted by* $G^d = (V^d, E^d)$*. Then the directed dual of G is a directed graph denoted by* $G^* = (V^*, E^*)$ *such that*

$$E^* = \{(u,v) \mid \{u,v\} \in E^d\}.$$

We are now ready to mention the main lemma from [MN95].

Lemma 11.3.1 ([MN95])**.** *Suppose* (G,c) *is a zero-demand graph. There exists a flow in* (G,c) *if and only if the directed dual* G^* *contains no negative weight cycle with respect to weights* c*.*

```
  Input   : A capacity-demand graph $(G,c,d)$
  Promise: $\sum_v d(v) = 0$
  Output  : A pseudo-flow in $(G,c,d)$
1 Construct a spanning tree $T$ in $G$;
2 For every edge $\{u,v\} \notin T$, set $f'(u,v) = 0$;
3 For each edge $\{u,v\} \in T$, deleting the edge $\{u,v\}$ separates the tree $T$ into two
    sub-trees, denoted as $T_u$ (sub-tree containing $u$) and $T_v$ (sub-tree containing $v$).
    Set $f'(u,v) = \sum_{w \in T_u} d(w)$;
```

Algorithm 8: `MN-Pseudo-Flow` [MN95]

Proof of Part 1 of Theorem 11.4. We can construct pseudo-flow in a capacity-demand graph having total demand zero, in log-space using the procedure `MN-Pseudo-Flow` (Algorithm 8) [DGKT12]. Now using the procedure `MN-Decision` (Algorithm 9), we reduce PerfectMatching (Decision) problem in planar bipartite (undirected) graph G to the problem of detecting negative weight cycle in the directed planar graph G^*, which can be solved in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space (by Corollary 11.1.1).

```
Input    : A planar bipartite (undirected) graph $G = (A \cup B, E)$
Output   : "Yes" if $G$ has a perfect matching; "No" otherwise
1 Construct a capacity-demand graph $(G, c, d)$ as follows: set $d(u) = 1, \forall_{u \in A}$ and
    $d(v) = -1, \forall_{v \in B}$. Also set $c(u, v) = 1$ and $c(v, u) = 0, \forall_{u \in A, v \in B}$;
2 Construct a pseudo-flow $f'$ in $(G, c, d)$;
3 Construct a zero-demand graph $(G, c - f')$;
4 Output Yes if the directed dual $G^*$ has no negative weight cycle with respect to
    weights $(c - f')$; Output No otherwise;
```

Algorithm 9: `MN-Decision` [MN95]

```
Input    : A planar bipartite (undirected) graph $G = (A \cup B, E)$
Promise: the directed dual $G^* = (V^*, E^*)$ has no negative weight cycle with
           respect to weights $(c - f')$
Output   : A perfect matching in $G$
1 Fix a vertex $s^* \in V^*$;
2 Set $f''(u^*, v^*) := dist^w_{G^*}(s^*, v^*) - dist^w_{G^*}(s^*, u^*), \forall_{u^* \in V^*}$;
3 Set $f = f'' + f'$;
4 For $u \in A$, $v \in B$ output "$u$ is matched with $v$" if and only if $f(u, v) = 1$;
```

Algorithm 10: `MN-Construction` [MN95]

Now observe in the Algorithm 10 that the construction of perfect matching in G boils down to the problem of finding the shortest distance $dist^w_{G^*}(u^*, v^*)$ between a pair of vertices $u^*, v^* \in V^*$ and we can do this in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space (by Theorem 11.1). Note that the space bound follows from the fact that the size of V^* is same as the number of faces in G, which is linear in number of vertices of G as G is planar. □

11.3.2 Constructing a Hall Obstacle

According to the Hall's Theorem [LP86], a bipartite (undirected) graph $G = (A \cup B, E)$ has a perfect matching if and only if $|A| = |B|$ and for every $S \subseteq A$, $|N(S)| \geq |S|$, where $N(S) := \{v \in B | \exists u \in A : (u, v) \in E\}$. A *hall-obstacle* in a bipartite graph $G = (A \cup B, E)$ is a set $S \subseteq A$ such that $|N(S)| < |S|$. We consider the following problems.

1. HallObs (Decision): given a bipartite (undirected) graph G, decide whether it contains a Hall-obstacle, and

2. HallObs (Construction): given a bipartite (undirected) graph G, construct a Hall-obstacle (if exists).

In this subsection, we mention the correspondence between the problem of constructing Hall-obstacle in a planar bipartite graph and the the problem of finding negative weight cycle in a planar graph. To do this we first restate some useful facts from [DGKT12].

Let $G = (A \cup B, E)$ be a planar bipartite (undirected) graph. Now consider the capacity-demand graph (G, c, d) and a pseudo-flow f' in it as defined in the procedure `MN-Decision` (Algorithm 9). Let C^* be a negative weight cycle in the directed dual

G^* with respect to weight $c - f'$. Let $(V_1 = A_1 \cup B_1, V_2 = A_2 \cup B_2)$ be the cut in G corresponding to C^*, where V_1 corresponds to the set of faces of G^* that are in the interior of C^* i.e., the vertices of G that are on one side of the cut corresponding to C^* and V_2 corresponds to the set of faces of G^* that are in the exterior of C^* i.e., the vertices of G that are on the other side of the cut corresponding to C^*. Since f' satisfies the property that for every edge $(u, v) \in E$, $f'(u, v) = -f'(v, u)$, $f'(C^*)$ decomposes into the sum of f's of the faces (in G^*) that are in the interior of C^*. Thus we have,

$$f'(C^*) = |A_1| - |B_1| \tag{11.1}$$

Lemma 11.3.2 ([DGKT12]). *Consider any edge $(a, b) \in (V_1, V_2)$ where $a \in A_1$, $b \in B_2$ and $c(a, b) = 1$. Then moving b from B_2 to B_1 does not increase the weight of the cut and the corresponding cycle in the dual with respect to the weights $c - f'$.*

Corollary 11.3.1 ([DGKT12]). *G^* has a negative weight cycle with respect to the weights $c - f'$ if and only if it has a negative weight cycle with respect to the weights $-f'$, and hence if and only if there exists a negative weight cycle with respect to the weights $cn^4 - f'$.*

Proof of Part 2 of Theorem 11.4. By Corollary 11.1.1, we can find negative weight cycle in G^* with respect to the weights $cn^4 - f'$ in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space. Since $N(A_1) \subseteq B_1$ and $|A_1| > |B_1|$ (see Equation 11.1), so the set A_1 forms a Hall-obstacle for G and this completes the proof. □

11.3.3 Deciding Even Perfect Matching

EvenPM denotes the following problem: given a graph G with each edge colored with either red or blue, decide whether there exists a perfect matching containing even number of red edges. Now consider the EvenPM problem in planar bipartite (undirected) graphs.

Proof of Theorem 11.5. Given a planar bipartite (undirected) graph $G = (V, E)$, first construct a perfect matching M in it, which can be done in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space (by Part 1 of Theorem 11.4). If M contains even number of red edges, then there is nothing to do. Otherwise, construct a weighted directed graph H with weight function w, as follows: H contains an edge (u, v) if and only if there exists $x \in V$ such that $\{u, x\} \in M$ but $\{x, v\} \notin M$. If the matching edge $\{u, x\}$ and the non-matching edge $\{x, v\}$ are of the same color, then set $w(u, v) = 0$; otherwise set $w(u, v) = 1$.

Claim 11.3.1. *There exists a perfect matching in G consisting of even number of red edges if and only if H contains an odd-weight cycle.*

Proof. Suppose H contains an odd-weight cycle. Now consider the corresponding portion of the graph in G, which is an even length cycle C consisting of alternating matched edge and non-matched edges. Now consider a new matching where every non-matched edge in C becomes matched and vice versa. The new matching is perfect because it does not affect the other part of matching in M and also matches every vertex in C. This new perfect matching contains even number of red edges as there are odd number of pair $\{u, x\}, \{x, v\}$ such that $\{u, x\} \in M$ but $\{x, v\} \notin M$ and they are of different colors.

For the converse direction, let us assume that M' is a perfect matching in G consisting of even number of red edges. If $M \cap M' \neq \phi$, then discard those common edges and now consider the sub-graph of G, say G', which contains an edge e if either $e \in M$ or $e \in M'$ but $e \notin (M \cap M')$. Now vertices in each of the connected component of G' are of degree 2 and thus each connected component is just a cycle. As M contains odd number of red edges and M' contains even number of red edges, so at least one of the cycles in G' contains odd number of red edges. Now if we consider the corresponding cycle in H (every cycle in G' corresponds to one cycle in H), then it must be of odd weight. □

Now observe that the process of construction of H is nothing but contraction of matched edges present in M on G and as G is a planar graph so the directed graph H is also planar. To check the presence of an odd-weight cycle in H, we construct another directed graph H' from H using the following process: replace every edge (x, y) having weight 0 by two edges (x, v_{xy}) and (v_{xy}, y) each with weight 1. It is easy to see that as H is a directed planar graph, so is the graph H'.

Claim 11.3.2. *H contains an odd-weight cycle if and only if H' contains an odd-weight cycle.*

Proof. If a cycle in H uses an edge (x, y) of weight 0, then there will be a corresponding cycle which will contain the portion $x \longrightarrow v_{xy} \longrightarrow y$ and as both the edges (x, v_{xy}) and (v_{xy}, y) have weight 1, so the parity of the resulting cycle will not change. So, if H contains an odd-weight cycle then H' also contains an odd-weight cycle.

Similar type of argument can be used to prove the converse direction as replacing the edges (x, v_{xy}) and (v_{xy}, y) in H' by the single edge (x, y) of weight 0 will result in a cycle of same parity in the graph H. □

Observe that the new graph H' contains $O(n)$ vertices as the original graph G is planar. As H' contains each edge of weight 1, so we can view this graph as an unweighted directed graph and then we can check the presence of odd length cycle in polynomial time and $O(n^{\frac{1}{2}+\epsilon})$ space (by Theorem 11.2.1) and this completes the proof. □

Chapter 12

Conclusion

In this part of this thesis, we make an incremental progress towards resolving the question "$\mathsf{NL} \subseteq \mathsf{SC}$?". For that purpose we consider directed reachability problem which is known to be NL-complete and show time-space upper bound for certain special graph classes. We show polynomial time and $\widetilde{O}(n^{2/3})$ space bound for H-minor-free graphs. The main ingredient of the proof is to efficiently construct a separator for this class of graphs. In general, if we have a space efficient construction of separator for any subclass of directed graphs then we can get similar time-space bound for that subclass [INP$^+$13]. However, it is not clear for which subclasses a separator exists. Certainly the subclasses for which we know that a separator exists, it is interesting to come up with a space efficient construction of such a separator for those subclasses. For example, we know the existence of similar kind of separators for various subclasses of chordal graphs [MW86] including interval and proper interval graphs [Pan15].

In Chapter 10 we provide a polynomial time algorithm that solves reachability problem for layered planar graphs using only $O(n^\epsilon)$ space for any $\epsilon > 0$. The immediate step after this is to extend the same time-space bound to the class of planar graphs and for that purpose it is sufficient to consider grid graphs instead of planar graphs because grid graph reachability is log-space reducible to planar reachability [ABC$^+$09]. The main hindrance in this direction is to get some clever "marking scheme" for the vertices that are already traversed while doing DFS. The question is whether we can come up with such a marking scheme or not.

Another question that we want to mention is whether we can use the similar approach as used in deciding reachability in layered planar graphs to establish the same time-space bound for the $\mathsf{RedBluePath}$ problem in the planar DAGs. Clearly this will put NL inside nSC and thus might be difficult. So as a first step one can consider layered planar DAGs instead of just planar DAGs and try to establish the same result. It is not at all clear how to extend techniques used in Chapter 10 in the $\mathsf{RedBluePath}$ problem for layered planar DAGs.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[ABC$^+$09] Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Planar and Grid Graph Reachability Problems. *Theory of Computing Systems*, 45(4):675–723, 2009.

[ACDN15] Manindra Agrawal, Diptarka Chakraborty, Debarati Das, and Satyadev Nandakumar. Dimension, Pseudorandomness and Extraction of Pseudorandomness. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*, volume 45 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 221–235, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[AD11] Tetsuo Asano and Benjamin Doerr. Memory-Constrained Algorithms for Shortest Path Problem. In *CCCG*, 2011.

[AHLM07] K. B. Athreya, J. M. Hitchcock, J. H. Lutz, and E. Mayordomo. Effective strong dimension, algorithmic information, and computational complexity. *SIAM Journal on Computing*, 37:671–705, 2007.

[AKNW14] Tetsuo Asano, David G. Kirkpatrick, Kotaro Nakagawa, and Osamu Watanabe. $\tilde{\mathrm{O}}(\sqrt{\mathrm{n}})$-Space and Polynomial-Time Algorithm for Planar Directed Graph Reachability. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 45–56, 2014.

[AL98] Eric Allender and Klaus-Jörn Lange. RUSPACE($\log n$) $\subseteq$ DSPACE ($\log^2 n$ / $\log\log n$). *Theory Comput. Syst.*, 31(5):539–550, 1998.

[AL06] K. B. Athreya and S. N. Lahiri. *Measure Theory and Probability Theory.* Springer Verlag, 2006.

[All07] E. Allender. Reachability problems: An update. *Computation and Logic in the Real World*, pages 25–27, 2007.

[ARZ98] Eric Allender, Klaus Reinhardt, and Shiyu Zhon. Isolation, Matching, and Counting: Uniform and Nonuniform Upper Bounds. *Journal of Computer and System Sciences*, 59:181, 1998.

[BBRS92] Greg Barnes, Jonathan F. Buss, Walter L. Ruzzo, and Baruch Schieber. A sublinear space, polynomial time algorithm for directed s-t connectivity. In *Structure in Complexity Theory Conference, 1992., Proceedings of the Seventh Annual*, pages 27–33, 1992.

[Bel58] Richard Bellman. On a routing problem. *Quart. Appl. Math.*, 16:87–90, 1958.

[Bil95] Patrick Billingsley. *Probability and Measure*. Wiley-Interscience, 3 edition, April 1995.

[BJLR91] Gerhard Buntrock, Birgit Jenner, Klaus-Jörn Lange, and Peter Rossmanith. Unambiguity and fewness for logarithmic space. In L. Budach, editor, *Fundamentals of Computation Theory*, volume 529 of *Lecture Notes in Computer Science*, pages 168–179. Springer Berlin Heidelberg, 1991.

[BLR90] Giuseppe Di Battista, Wei-Ping Liu, and Ivan Rival. Bipartite Graphs, Upward Drawings, and Planarity. *Inf. Process. Lett.*, 36(6):317–322, 1990.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, November 1984.

[BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In Sanjeev Arora, Klaus Jansen, Jos D. P. Rolim, and Amit Sahai, editors, *RANDOM-APPROX*, volume 2764 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2003.

[BT87] Giuseppe Di Battista and Roberto Tamassia. Upward Drawings of Acyclic Digraphs. In *Graph-Theoretic Concepts in Computer Science, International Workshop, WG '87, Kloster Banz/Staffelstein, Germany, June 29 - July 1, 1987, Proceedings*, pages 121–133, 1987.

[BTV09] Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed Planar Reachability Is in Unambiguous Log-Space. *ACM Transactions on Computation Theory*, 1(1):1–17, 2009.

[CGH$^+$85] Benny Chor, Oded Goldreich, Johan Håstad, Joel Freidmann, Steven Rudich, and Roman Smolensky. The bit extraction problem or t-resilient functions, 1985.

[Coo79] S.A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space. In *Proceedings of the eleventh annual ACM Symposium on Theory of Computing*, pages 338–345. ACM, 1979.

[Cov74] T. Cover. Universal gambling schemes and the complexity measures of Kolmogorov and Chaitin. Technical Report 12, Stanford University Department of Statistics, October 1974.

[CPT$^+$14] Diptarka Chakraborty, Aduri Pavan, Raghunath Tewari, N. V. Vinodchandran, and Lin F. Yang. New time-space upperbounds for directed reachability in high-genus and H-minor-free graphs. In *34th International Con-*

ference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India, pages 585–595, 2014.

[CR80] Stephen A. Cook and Charles Rackoff. Space Lower Bounds for Maze Threadability on Restricted Machines. *SIAM J. Comput.*, 9(3):636–652, 1980.

[CSV84] Ashok K. Chandra, Larry J. Stockmeyer, and Uzi Vishkin. Constant Depth Reducibility. *SIAM J. Comput.*, 13(2):423–439, 1984.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[CT15a] Diptarka Chakraborty and Raghunath Tewari. An $O(n^{\epsilon})$ space and polynomial time algorithm for reachability in directed layered planar graphs. In *Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, pages 614–624, 2015.

[CT15b] Diptarka Chakraborty and Raghunath Tewari. Simultaneous time-space upper bounds for certain problems in planar graphs. *CoRR*, abs/1502.02135, 2015.

[CT15c] Diptarka Chakraborty and Raghunath Tewari. Simultaneous time-space upper bounds for red-blue path problem in planar DAGs. In *WALCOM: Algorithms and Computation - 9th International Workshop, WALCOM 2015, Dhaka, Bangladesh, February 26-28, 2015. Proceedings*, pages 258–269, 2015.

[Das14] Debarati Das. Characterization of non-pseudorandomness. Master's thesis, Indian Institute of Technology Kanpur, 2014.

[DGKT12] S. Datta, A. Gopalan, R. Kulkarni, and R. Tewari. Improved bounds for Bipartite Matching on surfaces. In *Prooceedings of 29th Symposium on Theoretical Aspects of Computer Science (STACS)*, 2012.

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[DKLM10] Samir Datta, Raghav Kulkarni, Nutan Limaye, and Meena Mahajan. Planarity, Determinants, Permanents, and (Unique) Matchings. *ACM Trans. Comput. Theory*, 1(3):10:1–10:20, March 2010.

[DLN$^+$09] Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar Graph Isomorphism is in Log-Space. In *Annual IEEE Conference on Computational Complexity*, pages 203–214, 2009.

[DNTW09] Samir Datta, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Graph Isomorphism for $K_{3,3}$-free and K_5-free graphs is in Log-space. In *FSTTCS*, pages 145–156, 2009.

[EPA99] Jeff Edmonds, Chung Keung Poon, and Dimitris Achlioptas. Tight Lower Bounds for st-Connectivity on the NNJAG Model. *SIAM J. Comput.*, 28(6), 1999.

[FGT16] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 754–763, 2016.

[For56] Lester R. Ford Jr. Network Flow Theory. *Santa Monica, California: RAND Corporation*, pages P–923, 1956.

[G06] Sunil Kumar G. Characterization of randomness using betting games. Master's thesis, Indian Institute of Technology Kanpur, 2006.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques.* Cambridge University Press, 2001.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective.* Cambridge University Press, 2008.

[GRS04] Ariel Gabizon, Ran Raz, and Ronen Shaltiel. Deterministic extractors for bit-fixing sources by obtaining an independent seed. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 394–403, 2004.

[GT95] Ashim Garg and Roberto Tamassia. Upward planarity testing. *Order*, 12(2):109–133, 1995.

[HILL99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[Hita] J. M. Hitchcock. Effective Fractal Dimension Bibliography, http://www.cs.uwyo.edu/ ∼jhitchco/bib/dim.shtml (current April, 2011).

[Hitb] J. M. Hitchcock. Resource Bounded Measure - Bibliography, http://www.cs.uwyo.edu/ ∼jhitchco/bib/rbm.shtml (current April, 2011).

[Hit03] J. M. Hitchcock. Fractal dimension and logarithmic loss unpredictability. *Theoretical Computer Science*, 304(1–3):431–441, 2003.

[Hit04] John M. Hitchcock. Fractal dimension and logarithmic loss unpredictability, 2004.

[HLR07] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 169–186, 2007.

[HRV10] Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 437–446, 2010.

[INP$^+$13] T. Imai, K. Nakagawa, A. Pavan, N.V. Vinodchandran, and O. Watanabe. An $O(n^{1/2+\epsilon})$-Space and Polynomial-Time Algorithm for Directed Planar Reachability. In *Computational Complexity (CCC), 2013 IEEE Conference on*, pages 277–286, 2013.

[IW96] Russell Impagliazzo and Avi Wigderson. P=BPP unless E has sub-exponential circuits: Derandomizing the xor lemma (preliminary version). In *In Proceedings of the 29th STOC*, pages 220–229. ACM Press, 1996.

[KKR08] Sampath Kannan, Sanjeev Khanna, and Sudeepa Roy. STCON in Directed Unique-Path Graphs. In Ramesh Hariharan, Madhavan Mukund, and V Vinay, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 2 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 256–267, Dagstuhl, Germany, 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[KMW09] Philip N. Klein, Shay Mozes, and Oren Weimann. Shortest paths in directed planar graphs with negative lengths: a linear-space $O(n \log^2 n)$-time algorithm. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 236–245, 2009.

[KT05] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[Kul11] Raghav Kulkarni. On the Power of Isolation in Planar Graphs. *TOCT*, 3(1):2, 2011.

[KUW85] Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a Perfect Matching is in Random NC. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 22–32, 1985.

[KW11] Ken-ichi Kawarabayashi and Paul Wollan. A simpler algorithm and shorter proof for the graph minor decomposition. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 451–458, 2011.

[KZ03] Jesse Kamp and David Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. In *In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 92–101, 2003.

[LP84] Andrea S. LaPaugh and Christos H. Papadimitriou. The even-path problem for graphs and digraphs. *Networks*, 14(4):507–513, 1984.

[LP86] L. Lovasz and M.D. Plummer. Matching Theory. 1986.

[LRVW03] Chi-Jen Lu, Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Extractors: optimal up to constant factors. In Lawrence L. Larmore and Michel X. Goemans, editors, *STOC*, pages 602–611. ACM, 2003.

[Lut03a] J. H. Lutz. The dimensions of individual strings and sequences. *Information and Computation*, 187:49–79, 2003. Preliminary version appeared as [?].

[Lut03b] Jack H. Lutz. Dimension in complexity classes. *SIAM J. Comput.*, 32(5):1236–1259, 2003.

[Lut11] Jack H. Lutz. A divergence formula for randomness and dimension. *Theor. Comput. Sci.*, 412(1-2):166–177, 2011.

[MN95] Gary L. Miller and Joseph Naor. Flow in planar graphs with multiple sources and sinks. *SIAM Journal on Computing*, 24:1002–1017, 1995.

[MT01] Bojan Mohar and Carsten Thomassen. Graphs on surfaces. 2001. *Johns Hopkins Stud. Math. Sci*, 2001.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching Is as Easy as Matrix Inversion. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 345–354, 1987.

[MW86] Clyde L Monma and Victor K Wei. Intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 41(2):141 – 181, 1986.

[Ned99] Zhivko P. Nedev. Finding an Even Simple Path in a Directed Planar Graph. *SIAM J. Comput.*, 29:685–695, October 1999.

[Nis95] Noam Nisan. RL $\subseteq$ SC. In *In Proceedings of the Twenty Fourth Annual ACM Symposium on Theory of Computing*, pages 619–623, 1995.

[NT99] Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. *J. Comput. Syst. Sci.*, 58(1):148–173, 1999.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[NZ93] Noam Nisan and David Zuckerman. More deterministic simulation in logspace. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 235–244, 1993.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52:43–52, 1996.

[Pan15] Bhawani Sankar Panda. Separator theorems for interval graphs and proper interval graphs. In *Algorithms and Discrete Applied Mathematics - First International Conference, CALDAM 2015, Kanpur, India, February 8-10, 2015. Proceedings*, pages 101–110, 2015.

[Poo93] Chung Keung Poon. Space Bounds for Graph Connectivity Problems on Node-named JAGs and Node-ordered JAGs. In *FOCS*, pages 218–227, 1993.

[PT97] János Pach and Géza Tóth. Graphs Drawn with Few Crossings per Edge. *Combinatorica*, 17(3):427–439, 1997.

[PY82] Christos H. Papadimitriou and Mihalis Yannakakis. The Complexity of Restricted Spanning Tree Problems. *J. ACM*, 29(2):285–309, April 1982.

[Rag11] Nikhil Raghu. Quantifying pseudorandomness of distributions using betting games. Technical report, Indian Institute of Technology Kanpur, 2011.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4), 2008.

[RS03] Neil Robertson and P. D. Seymour. Graph Minors. XVI. Excluding a Nonplanar Graph. *J. Comb. Theory Ser. B*, 89(1):43–76, September 2003.

[RTS00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13:2000, 2000.

[RTV06] Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *In Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC 06)*, pages 457–466, 2006.

[Sav70] Walter J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci.*, 4:177–192, 1970.

[Sch71] C.P. Schnorr. A unified approach to the definition of random sequences. *Mathematical systems theory*, 5(3):246–258, 1971.

[Sha02] Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.

[SV12] Derrick Stolee and N. V. Vinodchandran. Space-Efficient Algorithms for Reachability in Surface-Embedded Graphs. In *IEEE Conference on Computational Complexity*, pages 326–333, 2012.

[TV00] Luca Trevisan and Salil P. Vadhan. Extracting randomness from samplable distributions. In *FOCS*, pages 32–42. IEEE Computer Society, 2000.

[TV12] Raghunath Tewari and N. V. Vinodchandran. Green's theorem and isolation in planar graphs. *Inf. Comput.*, 215:1–7, 2012.

[TW09] Thomas Thierauf and Fabian Wagner. Reachability in $K_{3,3}$-free Graphs and K_5-free Graphs is in Unambiguous Log-Space. In *17th International Conference on Foundations of Computation Theory (FCT)*, Lecture Notes in Computer Science 5699, pages 323–334. Springer-Verlag, 2009.

[Vin14] N. V. Vinodchandran. Space Complexity of the Directed Reachability Problem over Surface-Embedded Graphs. Technical Report TR14-008, I, 2014.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity - A Uniform Approach.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.

[VZ12] Salil P. Vadhan and Colin Jia Zheng. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 817–836, 2012.

[Wag37] K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen*, 114(1):570–590, 1937.

[Wig92] Avi Wigderson. The complexity of graph connectivity. *Mathematical Foundations of Computer Science 1992*, pages 112–132, 1992.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 80–91, Washington, DC, USA, 1982. IEEE Computer Society.

Index