RANDOMNESS AS A COMPUTATIONAL RESOURCE: ISSUES IN EFFICIENT COMPUTATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Suresh Chari

August 1994

RANDOMNESS AS A COMPUTATIONAL RESOURCE: ISSUES IN EFFICIENT COMPUTATION

Suresh Chari, Ph.D.
Cornell University 1994

Probabilistic methods have become an integral part of theoretical computer science. Typically, the use of randomization leads to solutions which are simple, elegant and more efficient than deterministic algorithms. A new research paradigm in randomized computation is to treat random bits as a resource and design algorithms which use random bits efficiently. The motivations for this approach are that algorithms which use fewer random bits are easier to implement. Also, algorithms using very few random bits can be made deterministic. For several problems all known efficient deterministic algorithms are obtained by this process of derandomization.

We design new probabilistic techniques that are used to develop randomness–efficient algorithms for combinatorial optimization problems. The first problem we consider is the abstract problem of isolating one of a possibly exponentially sized set of feasible solutions to the input instance. This tool has been used previously to design algorithms for problems such as perfect matching in graphs, basic problems on matroids and designing random reductions. We design a new technique to solve this problem that is more efficient

in the usage of random bits. Using this we derive randomness efficient solutions for the various applications. The randomness complexity of our method is parametrized in terms of the number of feasible solutions. As corollaries we obtain deterministic solutions for special cases of these problems. This new randomness–efficient technique unifies all previously known results for these applications and in many cases gives more efficient algorithms. We prove a strong lower bound which proves that our technique is optimal in the usage of random bits.

Another important tool is the concept of random sources whose distribution approximates that of a truly random source. We present a new construction of such sources that is specifically tailored to be easily combined with the algorithmic design method of conditional probabilities. Using this we derive improved deterministic parallel algorithms for several combinatorial optimization problems. We investigate the problem of constructing small sample spaces which satisfy given constraints. For several sets of constraints we present improved constructions of such sample spaces. These have obvious applications to randomness–efficient computation and obtaining deterministic algorithms.

Biographical Sketch

Suresh Chari was born in the teeming coastal city of Madras, India. He is deeply indebted to his parents for rescuing him from the oppressive weather at Madras by moving to the pleasant and hospitable city of Bangalore, India. After many enjoyable years there he went to study Computer Science at the Indian Institute of Technology in Kanpur, India. After a detour through The Ohio State University, he arrived at Cornell in 1989. In 1992 he was awarded a Master's degree in Computer Science. After promising to finish many times and threatening to quit at least ten times he will finally finish in the summer of 1994. Having spent five seemingly endless years in the frozen tundra on the shores of Lake Cayuga he is looking forward to enjoying the more clement climes of Southern California.

To the memory of Jayesh

Acknowledgements

My sincerest thanks go to Professor Juris Hartmanis for serving as the Chair of my thesis committee. From him I learned to look at the 'big picture' in my research. Very often, as a graduate student it is easy to get excited about technicalities and I am grateful to Juris for teaching me not to wallow in them for too long.

Thanks to Professors Anil Nerode, Ronitt Rubinfeld and David Shmoys for serving on my thesis Committee. David deserves special thanks for his extraordinary efforts at various times during my tenure at Cornell.

The theory group at Cornell has been a very enjoyable community to work in and I am grateful that I had an opportunity to meet the likes of Profs. Dexter Kozen and Sam Toueg. Thanks to Pankaj Rohatgi and Aravind Srinivasan with whom large parts of the work described in this thesis was done. I am grateful to Prof. Jin–yi Cai of Princeton/SUNY Buffalo for many interesting discussions during the two summers he visited Cornell. Diane Hernek deserves mention for suggesting the title of my dissertation.

Thanks to Devdatt Dubhashi, Steve Mitchell, Sendhil Mullainathan, Alessandro Panconesi, Pankaj Rohatgi, Aravind Srinivasan, Profs. David Shmoys, Paul Pedersen, and Ronitt Rubinfeld for their efforts in participating in the study group on Probabilistic

Methods during the Fall of 1992. Large parts of this thesis were directly influenced by the topics covered in the study group.

My years at Cornell have been made very pleasant by many friends both within the department and without. My heartfelt thanks to all the people who were instrumental in my surviving five years of graduate school: With apologies to any one I missed (and in no particular order!) I wish to thank Bhaskar, Jayesh, Priti, Pankaj, Silvia, Sandip, Mike K., Kjartan, Desh, Tushar, Sofoklis, Navin, Sonal, Mayan, Prasad, Sneha, ... for being there. Nandita deserves special thanks for introducing me to 'real' Latin American literature while I was at Ohio State. Thanks to Bill, Devdatt and Ida for having put up with my idiosyncrasies at the office.

Jan Batzer, Pauline Cameron and Rebekah Personius are exemplars at making insurmountable amounts of red–tape vanish while still maintaining a warm and cheerful outlook. But for them, I would have been harried by administrivia throughout my stay at Cornell.

Lastly and most importantly I would like to thank my parents who over the many years taught me the value of education and my brothers Ramesh and Srikumar for their constant encouragement.

Table of Contents

List of Figures

Chapter 1

Introduction

Probabilistic methods have emerged as the new paradigm in theoretical computer science over the past decade. Randomized algorithms for many problems are much easier to design than deterministic algorithms both in the sequential as well as the parallel domain. They are also usually much more efficient in the use of resources such as time and space. For many problems such as that of routing on fixed interconnection networks, one can prove that randomized algorithms are almost exponentially faster than any deterministic algorithm [Val82,VB81]. For several important number theoretic problems such as primality testing and factorization, the only known efficient algorithms are randomized.

Another application of probabilistic methods is in existence proofs: in many areas of theoretical computer science such proofs are almost always carried out by probabilistic methods. More recently, an investigation of the power of randomness in proof verification has led to many advances in cryptography such as zero knowledge protocols[GMR89, BoGKW88] and in resolving questions on how well several NP optimization problems, such as the maximum clique problem, can be approximated efficiently [ALMS92]. For an excellent introduction to the use of probabilistic methods we refer the reader to [ASE92] and [Kar91] for a survey on the randomized algorithms.

An area of research in probabilistic computation which has received considerable attention in the last few years is to treat random bits as a computational resource similar to time and space and to design randomness–efficient algorithms *i.e.* randomized algorithms which use as few and as low quality random bits as possible[Kar91]. One motivation for this approach is that when randomized algorithms are implemented then the source of truly random bits must be replaced by the output of a pseudo–random bit generator. Generating lots of good quality random bits this way is a costly and slow process. Therefore we want algorithms to use as few and as low quality random bits as possible. An important result exemplifying this philosophy is that of Karloff and Raghavan[KR88] who show that the standard quicksort algorithm works as well if the truly random source used to select a random pivot is replaced by the output of a linear pseudo–random generator. Other important results in this direction are that of Impagliazzo and Zuckerman [IZ89] on amplifying the success probability of randomized algorithms without using many extra random bits. In the recent results on efficient proof checking[ALMS92], the lower bounds on approximability of NP optimization problems depend crucially on proof verification protocols which use very few random bits.

However, the main motivation for this approach is that for a large class of combinatorial optimization problems the only known deterministic algorithms are obtained by the process of derandomization: First design a probabilistic algorithm which solves the problems and then show by analysis that this algorithm works as well with very restricted random sources. Since these restricted sources have very low entropy the probabilistic algorithm can be converted into a deterministic one by constructing the complete set of outputs of such a restricted random source[Lub88,NN90,AGHP92,ASE92] and then running the algorithm on each output of the source and taking a majority vote. Examples of combinatorial optimization problems for which the only known deterministic algorithms are obtained in this fashion are the set discrepancy problem, and parallel algorithms for approximating the set cover problem. Two common types of restricted sources usually considered are sources that guarantee limited independence of random variables and sources that approximate the distribution of independent binary random variables. The problem of efficiently constructing the complete set of outputs of such sources has been considered in the abstract and almost optimal constructions are known. Efficient constructions of small sample spaces which define a source that guarantees limited independence of random variables have been given by many authors [ABI86,Lub88, Jof74]. Constructions of sample spaces which approximate the distribution of independent binary random variables were first given by Naor and Naor[NN90] and later simpler constructions were given by Alon *et al.* [AGHP92]. Besides their application to obtain deterministic algorithms for the problems mentioned earlier, subsequently these constructions have become essential tools in many different algorithms and proofs[AN92, LY94].

In this thesis we adopt both the above approaches to obtain randomness–efficient and, in some cases, deterministic algorithms for several combinatorial optimization problems. In Chapter 2 we consider the problem of parallel algorithms for finding a perfect matching in graphs. This is a fundamental problem in combinatorial optimization and has connections to several other important problems such as network flows. While the matching problem has been fairly well solved in the sequential domain, a fundamental open question in the theory of parallel algorithms is whether this problem can be solved efficiently in parallel. This is perhaps the most notable example of a problem for which no parallel algorithms are known and yet the problem is not known to be complete for the class of polynomial time computations under reductions which use logarithmic space. The only known parallel algorithms for this problem are probabilistic[MVV87, KUW86,CRS93]. Deterministic parallel algorithms are known only for special classes of input graphs [GK87]. We develop a new randomness efficient algorithm for the general problem. The randomness–complexity of our algorithm is better than those of previously known algorithms and, more importantly, parametrized in terms of a bound on the number of perfect matchings in the input graph. As a corollary, this yields efficient deterministic parallel algorithms for special case of graphs with a polynomial bound on the number of perfect matchings. These deterministic algorithms are much more efficient than algorithms designed using other techniques. This technique extends to several generalizations and variants of the perfect matching problem and we obtain similar results for these problems. An important application, to sequential computation, of the improved technique that we develop in Chapter 2 is to obtain a new random reduction from languages in the class NP to the language of uniquely satisfiable Boolean formulas. A random reduction between two languages A and B indicates that A is not much harder than B in the sense that a randomized algorithm can 'bridge the gap' between A and B. We explore this application in detail in Chapter 3. We develop several alternate random reductions which are both simpler and more randomness efficient than previously known reductions. The general technique that is developed in chapter 2 thus unifies several results in the various fields to which it is applied. However, due to its abstract and general nature we are able to show a general lower bound which shows that the technique that we develop is optimal in its usage of random bits. This establishes that our perfect matching algorithm is optimal in its usage of entropy: any algorithm using a similar technique must have the same entropy as that of our algorithm. This lower bound rules out several possible approaches to obtaining deterministic parallel algorithms for the perfect matching problem.

We then consider the problem of efficiently generating the possible outputs of a source which approximates the distribution of independent binary random variables. In Chapter 4 we develop a new construction for this problem. The entropy of our source can, in general, be more than that of the approximate random sources developed earlier[NN90, AGHP92]. However our construction is specifically tailored so that it can be effectively combined with the classical algorithmic design method of conditional probabilities. We use this new construction to give new improved parallel algorithms for several combinatorial optimization problems such as the set discrepancy problem, the approximate set cover problem etc. The algorithms that we obtain are better than those obtained using the old constructions of approximate sources. In each application our method results in algorithms that are faster than previously known algorithms or match the running time of the best known algorithms with a significant reduction in the processor complexity.

Given the utility of these approximately independent distributions we consider the generalized problem of efficiently constructing other sources of small entropy which satisfy given constraints on the output random variables. For various sets of constraints, we give new constructions which are of lower entropy than those of previously known constructions. We also study the question of lower bounds on the entropy of sources which satisfy these constraints. Here again we obtain better lower bounds than those previously known.

Chapter 2

Efficient algorithms for Perfect Matching and its generalizations [1]

Given a graph G, a *matching* in G is a subset of the edges such that no two edges in the subset have a common endpoint. A *maximum matching* in G is a matching in G of maximum cardinality, and a *perfect matching* is a special type of maximum matching, in which each vertex is covered by an edge, *i.e.*, each vertex is *matched.* Finding a perfect matching, if one exists, in a graph is one of the fundamental problems in combinatorial optimization. Approaches to solving this problem have led to the formalization of the key concepts of complexity theory. The sequential complexity of the matching problem has been well-characterized (Lovász & Plummer [LP86]) and several efficient algorithms such as the algorithm of Micali and Vazirani[MV80] are known. However, a fundamental problem in the theory of parallel combinatorial algorithms is whether a perfect matching

[1] The results presented in this chapter are joint work with Pankaj Rohatgi and Aravind Srinivasan and a preliminary version appears in the Proceedings of the 25^{th} Annual Symposium on the Theory of Computing.

can be found or even detected efficiently in parallel: if the maximum matching problem in the class NC *i.e.* if it can be solved polylogarithmic time using a polynomial number of processors? Due to the fundamental nature of matching, a considerable effort has been devoted towards developing parallel algorithms for the perfect matching problem. One approach for the general matching problem has been to focus on obtaining parallel algorithms which use a polynomial number of processors but run in sublinear as opposed to polylogarithmic time (see for example Goldberg, Plotkin & Vaidya [GPV93], Vaidya [Vai90], Grover [Gro92] and Goldberg, Plotkin, Shmoys & Tardos [GPST92] for such algorithms for the case of bipartite graphs). The only known NC algorithms for the matching problem are for special classes of graphs such as for co–comparability graphs (Kozen, Vazirani & Vazirani [KVV85]), strongly chordal graphs (Dahlhaus & Karpinski [DK86]), graphs with polynomially many perfect matchings (Grigoriev & Karpinski [GK87]), and planar bipartite graphs (Miller & Naor [MN89]). The only known algorithms for the general problem which run in polylogarithmic time with a polynomial number of processors also use randomness: both Monte Carlo RNC algorithms (Karp, Upfal & Wigderson [KUW86], Mulmuley, Vazirani & Vazirani [MVV87]) and Las Vegas RNC algorithms (Karloff [Kar86]) have been developed.

In this chapter, we investigate the *parallel randomness complexity* of perfect matching and related problems, *i.e.*, the amount of randomness required to solve them in parallel. Our results include an RNC^2 algorithm for perfect matching which uses $O(\log Z + \log n)$ random bits where Z is any *given* upper bound on $|\mathrm{Mat}(G)|$, the number of perfect matchings in the graph G. When the upper bound Z is polynomial, this uses $O(\log n)$ bits which can be derandomized to yield an NC^2 algorithm. This improves and generalizes the result of Grigoriev & Karpinski [GK87] who give an NC^3 algorithm when Z is polynomially bounded. Since the number of perfect matchings in a graph on n vertices with m edges is upper bounded by $(2m/n)^n$, the worst case randomness complexity of our algorithm is $O(n \log(m/n))$ which improves on the previous bound of $O(m \log n)$ due to Mulmuley, Vazirani & Vazirani [MVV87]. Thus by linking the randomness complexity of this problem to the number of perfect matchings, we unify and generalize previous results, while also improving on them. In special cases, *e.g.*, $K_{3,3}$–free graphs, $|\mathrm{Mat}(G)|$ can be computed in NC^2 (see for example Vazirani [Vaz89] and also the work of Kastelyn[Kas67] and Little [Lit74]), we get algorithms which use exactly $O(\log(|\mathrm{Mat}(G)|) + \log n)$ random bits; in general, if no good upper bound on $|\mathrm{Mat}(G)|$ can be found, our results yield an RNC^3 algorithm to find a perfect matching which uses at most $O(\log |\mathrm{Mat}(G)|)$ random bits with high probability, if $|\mathrm{Mat}(G)| > 0$. This is a significant reduction in the randomness complexity if $0 < |\mathrm{Mat}(G)| \ll (2m/n)^n$. All of our results are obtained from a *randomness–optimal* generalization of the Isolating Lemma, a tool used to isolate a perfect matching in the RNC algorithm of Mulmuley, Vazirani and Vazirani[MVV87]. This abstract and powerful tool has several applications, including algorithms for basic problems on linearly representable matroids, variants of matching and random reductions in structural complexity theory. Our generalization results in randomness–efficiency in these settings as well.

Given a set $S = \{x_1, x_2, \ldots, x_N\}$ and an *unknown* family $\mathcal{F} \subseteq 2^S$, the Isolating Lemma of [MVV87] shows that if the x_i's are *independently* assigned weights uniformly at random from the range $\{1, 2, \ldots, 2N\}$, then with probability at least $\frac{1}{2}$, the minimum weight set in $\mathcal{F}$ is *unique*. This weight assignment scheme clearly requires $\Theta(N \log N)$

random bits. A question left open in the original paper was whether the assumption that the weights be assigned independently is necessary. Our scheme generalizes this as follows: given an upper bound Z on $|\mathcal{F}|$, we assign polynomially bounded weights to the x_i's using only $O(\log Z + \log N)$ random bits and achieve the same result. In the worst case ($Z = 2^N$), our scheme needs $O(N)$ random bits as compared to $\Theta(N \log N)$; more importantly, for smaller Z, we get much better randomness complexity. Also, since even in the worst case we assign polynomial weights to N different random variables using $O(N)$ random bits, the weight assignments are not independent, thus answering the open question of [MVV87] in the affirmative. Our scheme provides an explicit construction of a collection of $(NZ)^{O(1)}$ weight assignments such that, for every family $\mathcal{F}$ of size at most Z, at least one (in fact, at least 50 percent) of these assignments makes the minimum weight subset in $\mathcal{F}$ unique. Such a collection of size $O(NZ)$ is guaranteed to exist by applying Adleman's technique [Adl78] of constructing a small set of witnesses for probabilistic computation to the original Isolating Lemma, and thus this is one of the few instances where Adleman's result can be made constructive with only a polynomial blowup in size. Furthermore, we also show that our weight assignment scheme is optimal by proving a matching lower bound for this generalization, *i.e.*, any scheme which assigns polynomial weights to the x_i's must use $\Omega(\log Z + \log N)$ random bits even to achieve isolation with only *nonzero* probability. This indicates that any scheme based on isolation which treats the family of perfect matchings as nothing but an abstract collection of subsets must use $\Omega(n)$ random bits in the worst case. While this only establishes a lower bound for this specific weight assignment schema, in the next chapter we shall show that the same lower bound holds for a much more generalized class of approaches.

This randomness–efficient generalization of the isolation process can be directly plugged into the settings where the original Isolating Lemma has been used to obtain randomness efficiency. Besides the new algorithm for perfect matching we get randomness–efficient algorithms for variants of perfect matching such as the exact matching and maximum matching problems. We get an RNC^2 algorithm to find an exact matching in a graph G which uses $O(\log X)$ random bits, where X is a *given* upper bound on the number of exact matchings and a similar result holds for maximum matchings. Matroid intersection and matroid matching are generalizations of bipartite and general graph matching, and the Isolating Lemma has been used to obtain RNC^2 algorithms for these basic problems on linearly representable matroids (Narayanan, Saran & Vazirani [NSV92]). Here, a careful inspection of their algorithm yields good bounds on the size of the family on which the Isolating Lemma operates, and thus we obtain significant savings in the number of random bits required. The randomness complexities of our algorithms for matroids, when specialized to the case of matching, yield precisely our results on matching. In particular, we give NC^2 algorithms for these problems on matroids under certain restrictions which in the case of graph matching correspond exactly to a polynomial bound on $|\mathrm{Mat}(G)|$. This generalizes the results of [GK87] to matroids and also generalizes a result due to Tiwari[Tiw87].

Another important application of the Isolating Lemma is to obtain an alternative to the Valiant & Vazirani random reduction [VV86] from languages in the class NP to the language, USAT, of uniquely satisfiable formulas. Our generalization yields a new random reduction whose randomness complexity is logarithmic in the number of satisfying assignments. The worst case complexity matches the best–known bound of

$O(n)$ (Tarui [Tar91]) and also directly gives new proofs of the results that FewP, where the number of satisfying assignments is polynomially bounded, is contained in $\oplus P$ (Cai & Hemachandra [CH90]) and in $\mathrm{C_{=}P}$ (Köbler, Schöning, Toda & Toran [KSTT92]). We will explore this application in greater detail in the following chapter.

2.1 The Generalized Isolating Lemma

In this section we present our randomness–efficient generalization of the Isolating Lemma and also show its optimality by proving a matching randomness complexity lower bound for this problem. We start with a formal statement of the Isolating Lemma:

Definition A *set system* $(S, \mathcal{F})$ consists of a finite set $S = \{x_1, \ldots, x_N\}$ and a family $\mathcal{F}$ of subsets of S. A weight assignment $\vec{w} = \langle w_1, \ldots, w_N \rangle$ to the elements $x_1, \ldots, x_N$, extends naturally to sets in $\mathcal{F}$ with $\vec{w}(S_j) = \sum_{x_i \in S_j} w_i$.

Isolating Lemma [MVV87] *Let $(S, \mathcal{F})$ be any set system. If the elements x_i of S are assigned random weights* uniformly and independently *from $\{1, 2, \ldots, 2N\}$ then, with probability at least $1/2$, there will be a* unique *minimum weight set in the family $\mathcal{F}$.*

Two features make the Isolating Lemma widely applicable. First, the isolation process works for arbitrary and unknown families $\mathcal{F}$ and second, isolation is achieved by assigning only polynomial weights. For example, in the bipartite perfect matching problem where S is the set of edges and $\mathcal{F}$ is the collection of perfect matchings, the first feature allows isolation to be done in RNC without any knowledge of the family of perfect matchings and the second feature ensures that a matching can be found by inverting a matrix with *polynomial* sized entries as shown in the original paper[MVV87].

2.1.1 New Isolation Scheme

We prove a randomness–efficient generalization of the Isolating Lemma which also assigns polynomial weights and which depends only on any given upper bound Z on the size of the *unknown* family $\mathcal{F}$. The motivation for this generalization is that in several settings where the Isolating Lemma has been used to obtain randomized algorithms in the general case, we have deterministic solutions using completely different techniques when the number of solutions is small[GK87,Tiw87,CH90]. These deterministic solutions have been obtained by other techniques that are specialized to each case and our aim is to unify these two classes of results by carefully parametrizing the randomness complexity of the algorithms for the general case. To do this we prove the following generalization:

Lemma 2.1 [Generalized Isolating Lemma] *Let $(S,\ \mathcal{F})$ be any set system and let Z be a* given *upper bound on the size of the unknown family $\mathcal{F}$. There is a simple scheme which uses $O(\log Z + \log N)$ random bits to assign integer weights to the x_i's in the range $[0, N^7)$ such that with probability at least $1/4$, there is a* unique *minimum weight set in $\mathcal{F}$.*

Proof. We will outline a four-step process for assigning weights to the ground set elements and prove that this scheme has the desired properties. At step j, we denote the intermediate weight assigned in that step by $\vec{w}^{(j)}$.

To begin, since $\mathcal{F}$ is unknown, we *deterministically* assign very large weights in Step 1 so that every set in $\mathcal{F}$ gets a *distinct* weight.

Step 1: For each i, set $w_i^{(1)} = 2^i$.

Note that the weight of a subset of ground set elements in binary is exactly the characteristic vector of the subset. Hence, under $\vec{w}^{(1)}$, sets in $\mathcal{F}$ have *distinct* weights in $\{1, 2, \ldots, 2^{N+1}\}$. Clearly, if $Z \ll 2^{N+1}$, the same property should be obtainable with much lower weights. Since $\mathcal{F}$ is unknown, a deterministic strategy for reducing weights may fail; instead, we use the following randomized strategy.

Step 2: Choose m uniformly at random from $\{1, 2, \ldots, (2NZ^2)^2\}$. For each i, define $w_i^{(2)} = w_i^{(1)} \bmod m$.

Step 2 requires $O(\log Z + \log N)$ random bits to sample a random integer from the range $[1, (2NZ^2)^2]$. Under $\vec{w}^{(2)}$, the Z or fewer sets in $\mathcal{F}$ have weights in the interval $[0, \min(N(2NZ^2)^2, 2^{N+1})]$, which is a big improvement for small values of Z. We now claim that with good probability, these weights are also *distinct*.

Claim 2.1 *With probability at least* $1/2$*,* all *sets in* $\mathcal{F}$ *have* distinct *weights under the weight assignment* $\vec{w}^{(2)}$*.*

Proof. Suppose $\mathcal{F} = \{S_1, \ldots, S_k\}$ where $k \le Z$. Consider the (unknown) integer

$$I = \prod_{1 \le i < j \le k} (\ \vec{w}^{(1)}(S_i) - \vec{w}^{(1)}(S_j)\ ).$$

Clearly the properties of $\vec{w}^{(1)}$ ensure that $I \ne 0$ and since there are at most $\left({}_{Z2<Z^2} \right)$ terms in the product, $|I| < 2^{2NZ^2}$. The following number-theoretic proposition, together with the Chinese Remainder Theorem establishes that when m is chosen uniformly from $\{1, 2, \ldots, (2NZ^2)^2\}$, the probability that $m \not| I$ is at least $1/2$. Several versions of this proposition appear in the literature; this version is from [Tom91] and the constant 89 has been improved to 3 by Thrash [Thr93].

Proposition 2.1 *Let $L \geq 89$ and let S be any subset of $\{1, \ldots, L^2\}$ such that $|S| \geq \frac{1}{2}L^2$. Then, the least common multiple of the elements of S exceeds 2^L.*

We claim that if $m \not| I$, then all sets in $\mathcal{F}$ have *distinct* weights under $\vec{w}^{(2)}$. If not, then $\vec{w}^{(2)}(S_i) = \vec{w}^{(2)}(S_j)$ for some $i < j$. Then we have $m \mid (\vec{w}^{(1)}(S_i) - \vec{w}^{(1)}(S_j))$, which contradicts the fact that $m \not| I$. $\square$

Remark. We say that Step 2 *succeeds* if sets in $\mathcal{F}$ get distinct weights under $\vec{w}^{(2)}$. The success probability of this step can be boosted to any constant smaller than 1, since it follows from the results of de Bruijn [dB51] that for any fixed $p < 1$, there exist constants c_1 and c_2 such that if m is picked uniformly at random from $\{1, 2, \ldots, N^{c_1} Z^{c_2}\}$ in Step 2, then Step 2 will succeed with probability at least p.

Note that if Z is at most a polynomial N^c, then the weight of any ground set element can at most be N^{4c+3} and with probability at least $1/2$, *all* sets in $\mathcal{F}$ have *distinct* weights. Later, we use this strong condition to design an NC^2 algorithm to find *all* perfect matchings in graphs with at most $N^{O(1)}$ perfect matchings. For larger Z, the weights are still too big and in Steps 3 and 4 we reduce them further. Step 2 assigns q-bit weights to the x_i's where $q = min(N, \log m) \leq min(N, 4\log Z + 2\log(2N))$. Let $t = \lceil q/\log N \rceil$.

Step 3: For each i, write $w_i^{(2)}$ as q-bit number. Split these bits into t blocks of size $\log N$ bits each as shown. Let $b_{i,j}$ be the number in $[0, N-1]$ formed by the bits in block j.

$$w_i^{(2)} : \quad \overbrace{\boxed{b_{i,t-1}}\boxed{\cdots}\boxed{b_{i,1}}\underbrace{\boxed{b_{i,0}}}_{\log N}}^{q}$$

Let $w_i^{(3)}$ be the *linear form* $\sum_{j=0}^{t-1} b_{i,j} \cdot y_j$ over the variables $y_0, \ldots, y_{t-1}$.

Claim 2.2 *If Step 2 succeeds then the linear forms* $\vec{w}^{(3)}(S_j)$ *, where* $S_j \in \mathcal{F}$*, are all* distinct.

Proof: Assume that Step 2 succeeds, *i.e.*, that all the weights $\vec{w}^{(2)}(S_j)$ are *distinct*, where $S_j \in \mathcal{F}$. Note that each of the linear forms under $w_i^{(3)}$ *evaluated* with the substitution $y_k = 2^{k \log N}$, $0 \le k \le t-1$, is *exactly* the weight $w_i^{(2)}$. This implies that each $\vec{w}^{(3)}(S_j)$ evaluates to the *distinct* value $\vec{w}^{(2)}(S_j)$ at $y_k = 2^{k \log N}$, $0 \le k \le t-1$, which implies that the forms $\vec{w}^{(3)}(S_j)$'s must be distinct. $\square$

Each of the linear forms corresponding to the weight of a ground set element has coefficient of magnitude at most N and the coefficients of the weight of subsets are at most N^2. We will use this property in a crucial way in the analysis of the next and final step.

Step 4: Choose $r_0, \ldots, r_{t-1}$ uniformly and independently at random from $\{1, 2, \ldots, N^5\}$. For each i, set $w_i^{(4)}$ to be the evaluation of $w_i^{(3)}$ at $y_k = r_k$, $0 \le k \le t-1$.

We claim that $\vec{w}^{(4)}$ achieves the requirements of the Generalized Isolating Lemma. Clearly, Step 4 requires $5 \log N \times t = O(\log Z + \log N)$ random bits and since Step 2 has a similar randomness complexity, the overall procedure requires $O(\log Z + \log N)$ random bits. It is easy to check that each $w_i^{(4)}$ can be at most $[0, N^7)$. Since Step 2 succeeds with probability at least 1/2, by using $\mathcal{C} = \{\vec{w}^{(3)}(S_j) \mid S_j \in \mathcal{F}\}$ in the following proposition, we obtain that the weight assignment $\vec{w}^{(4)}$ achieves isolation with probability at least 1/4.

Proposition 2.2 *Let $\mathcal{C}$ be any collection of* distinct *linear forms over at most l variables $\vec{y} = y_0, y_1, \ldots, y_{t-1}$ with coefficients in $\{0, 1, \ldots, N^2 - 1\}$. Choose a random $\vec{r} = r_0, \ldots, r_{t-1}$ by assigning each r_i uniformly and independently from $\{1, 2, \ldots, N^5\}$. Then in the assignment $\vec{y} = \vec{r}$ there will be a* unique *linear form with minimum value, with probability at least* $1/2$.

Proof of Proposition: Our proof parallels that of [MVV87]; we define a variable y_i to be *singular* under an assignment $\vec{r}$ to the variables $\vec{y}$ if there exist two minimum valued linear forms in $\mathcal{C}$ under this assignment, having different coefficients of y_i. Thus

$$\text{Prob.[minimum valued form is not unique]} \;=\; \text{Prob.[} \exists i: \; y_i \text{ is singular]}$$

For each y_i, we will upper bound the probability that it is *singular.* Fix an i and assume that the variables $\vec{r}(\bar{i}) = r_0, \ldots, r_{i-1}, r_{i+1}, \ldots, r_{l-1}$ have been assigned the values $\vec{a}(\bar{i}) = a_0, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{l-1}$. Under this partial assignment the weight of each linear form is of the form $a + by_i$ where a is the *partial weight* of the linear form and the coefficient b is at most $N^2 - 1$. The family $\mathcal{C}$ can be partitioned into at most N^2 classes $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{N^2-1}$ based on the coefficient of y_i, *i.e.*, forms in $\mathcal{C}_j$ have j as the coefficient of y_i. By definition, y_i is *singular* under an assignment $\vec{r}$ if and only if there are two linear forms from two different classes in the partition which attain the minimum weight. Consider the probability that the choice of r_i makes y_i singular, conditioned on the partial assignment to the other variables. Let p_j be the minimum partial weight among all the forms in $\mathcal{C}_j$. Note that since all the forms in a particular class have the same coefficient of y_j in their weights, only the forms with minimum partial weight in each class can possibly attain the minimum overall weight. Thus the probability that r_i makes y_i *singular* is exactly the probability that the minimum valued element in the list $\mathcal{D} = [p_0, p_1 + r_i, p_2 + 2r_i, \ldots, p_{N^2-1} + (N^2-1)r_i]$ is not unique. This is at most the probability that under a random r_i the values of elements in $\mathcal{D}$ are not all distinct. For each pair of elements in $\mathcal{D}$, *at most one* choice of r_i makes their values equal. Thus

$$\begin{aligned}
\text{Prob.}[\ y_i \text{ is singular} \mid \vec{r}(\bar{i}) = \vec{a}(\bar{i})\ ] &\leq \text{Prob.}[\ \exists l, m:\ p_l + l \times r_i = p_m + m \times r_i\ ] \\
&\leq \binom{N^2}{2} \times \text{Prob.}[\ p_l + l \times r_i = p_m + m \times r_i\ ] \\
&\leq \binom{N^2}{2} \times \frac{1}{N^5} = \frac{1}{2N}
\end{aligned}$$

Since r_i is chosen independently of the other variables $\vec{r}(\bar{i})$ the probability that the assignment makes y_i singular is at most $\frac{1}{2N}$. Since this holds for each variable, and there are at most $t < N$ variables we have

$$\begin{aligned}
\text{Prob.}[\ \text{minimum valued form in } \mathcal{C} \text{ is unique}\ ] &\geq 1 - \text{Prob.}[\ \exists j\ \ y_j \text{ is singular}] \\
&\geq 1 - t \times \frac{1}{2N} > \frac{1}{2}
\end{aligned}$$

$\square$

Since each of the four steps is elementary the entire weight assignment scheme can be implemented in NC^1. Finally, we note that the Generalized Isolating Lemma provides an *explicit construction* of $(NZ)^{O(1)}$ many weight assignment functions for the ground set $S = \{x_1, x_2, \ldots, x_N\}$, such that for any set system $(S, \mathcal{F})$ with $|\mathcal{F}| \leq Z$, at least half of these functions will produce a unique minimum weight set in $\mathcal{F}$. For a given ground set S, the number of set systems $(S, \mathcal{F})$ with $|\mathcal{F}| \leq Z$ is $\sum_{i=1}^{Z} \binom{2^N}{i}$; hence, the technique of Adleman [Adl78] combined with the original isolating lemma shows *nonconstructively* that there exists a set of $O(\log(\sum_{i=1}^{Z} \binom{2^N}{i})) = O(NZ)$ many weight assignment functions such that for any set system $(S, \mathcal{F})$ with $|\mathcal{F}| \leq Z$, at least *one* of these functions will produce a unique minimum weight set in $\mathcal{F}$. Thus our generalization is one of the few places where one can constructivize Adleman's bound without a significant blowup in size.

The above generalization clearly assigns weights to the ground set elements which are not independent. Thus the scheme is more randomness–efficient than the original weight assignment scheme. However, the original scheme possesses some properties which the above weight assignment scheme lacks. Under the original weight assignment scheme, assuming that no subset in the family $\mathcal{F}$ is contained in another, the probability that any particular element attains the unique minimum value is nonzero. However the same can not be said about the above scheme. For instance, consider the subsets $S_1 {=} \{x_1\}$ and the subset $S_2 {=} \{x_2\}$. It can be easily verified that under all choices of random bits for which an unique minimum weight subset is produced, the weight of S_2 is always larger than that of S_1. Another property possessed by the original scheme is that the difference between the minimum weight subset and the subsets with next highest weight can be lower bounded. It is not obvious that the same can be said of the new weight assignment scheme. Thus non independence of weights destroys the symmetry in weight assignments to the subsets.

2.1.2 A Lower bound for the Isolation problem

In this section we establish a matching lower bound of $\Omega(\log Z + \log N)$ random bits on the randomness complexity of the generalized isolation process, and in fact for the following weaker problem, thus showing that the randomness complexity of our weight assignment scheme is optimal to within a constant factor.

Definition 1 Generalized Isolation *Let $S = \{x_1, \ldots, x_N\}$ and let k be a constant. Assign weights to the elements of S in the range $[1, N^k]$ such that for any family $\mathcal{F} \subseteq 2^S$ of size at most Z, the minimum weight set in $\mathcal{F}$ is unique with* nonzero *probability.*

It appears that there is an obvious lower bound for the problem: Since there are up to Z solutions we need to address this space of Z solutions we need at least $\log Z$ random bits. However, this entropy argument is incorrect since it assumes that every solution of the space must be output on some path as the unique minimum weight solution. In fact, this is not the case for the generalization that we have shown.

We first prove a lower bound of $\Omega(\log Z)$ random bits for any randomized scheme for the above problem. Each path of such a randomized algorithm defines a weight assignment f to the x_i's. The following theorem shows that at least $t+1$ different weight assignments are needed in order to achieve isolation in all possible families of size at most Z, where $t = \min(\lfloor \frac{Z}{2} \rfloor, \lfloor \frac{2^N - 3}{N^{k+1}} \rfloor)$. Hence any randomized scheme for generalized isolation must use $\Omega(\log t) = \Omega(\log Z)$ random bits so that at least t different weight assignments can occur over all possible coin tosses.

Theorem 2.1 *Let $f_1, f_2, \ldots, f_t$ be any collection of weight assignments which assign weights in the range $[1, B]$ to the elements of S. If $t \leq \min(\frac{Z}{2}, \frac{2^N - 3}{NB})$, then there exists* *a family $\mathcal{F}$ of subsets of S with $|\mathcal{F}| \leq 2t \leq Z$, such that the minimum weight subset of $\mathcal{F}$ is not unique under any of these assignments.*

Proof. Given any t weight assignments we explicitly construct the family $\mathcal{F}$ as follows. First, for each f_i, we compute the "histogram" of the weights of the subsets of S, *i.e.*, for each nonempty subset X we place a mark above the weight $f_i(X)$ in the histogram for f_i; one distinct mark is made for each such subset X.

The weight of any subset is in the range $[1, NB]$. Initialize, for each i, a pointer p_i to 1 in the histogram for f_i. The initial configuration of the pointers and the histogram are shown in Figure 2.1. The pointers p_i advance according to the following rules:

- If there are no marks on weight p_i in the histogram of f_i, then advance p_i to $p_i + 1$.
- If there is exactly one mark on weight p_i which corresponds to some subset X, remove the marks corresponding to X from *all* the histograms and advance p_i to $p_i + 1$.

If more than one pointer can advance, we choose one among them arbitrarily. We continue this process until the pointers cannot move any further. First we argue that not all marks can be removed since every time a mark is removed, one of the pointers advances and hence the potential function $\Psi = \sum_i (p_i - 1)$ always increases. Since each p_i is bounded by $NB + 1$, Ψ is bounded by tNB. By assumption $tNB \leq 2^N - 3$, and hence when the pointers come to a halt there are at least two marks corresponding to nonempty subsets X_{i1} and X_{i2} on weight p_i in the histogram of f_i, for each i. The family $\mathcal{F} = \{X_{i1}, X_{i2} | \, 1 \leq i \leq t\}$ has size at most $2t \leq Z$ and for each i, there are two nonempty

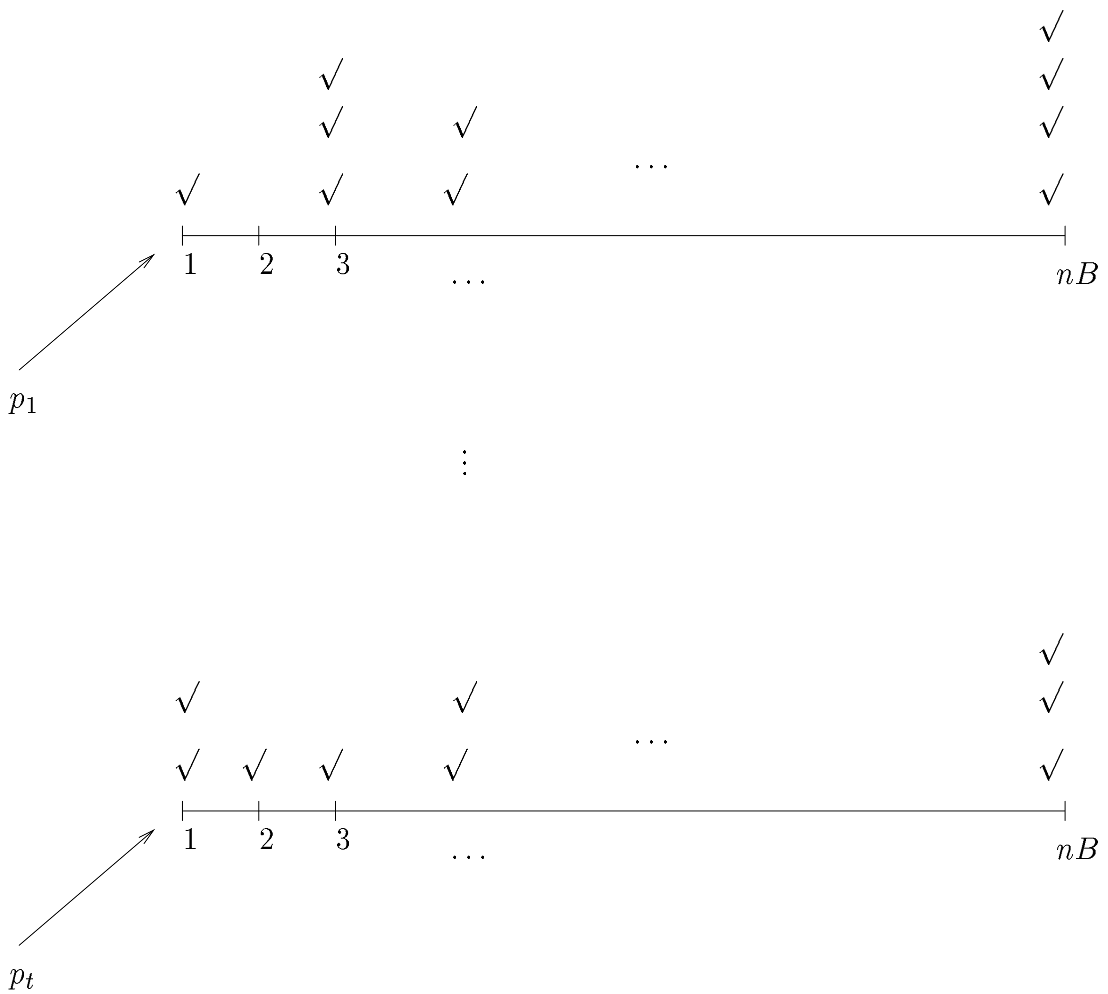

Figure 2.1: A possible set of histograms under the t weight assignments and the initial configuration. Each mark ($\surd$) corresponds to some nonempty subset of S.

minimum weight subsets, X_{i1} and X_{i2}, of minimum weight p_i, under the assignment f_i. $\square$

Also when $B = n^{O(1)}$ and $t = o(\frac{N}{\log N})$, we can construct two sets which have the same weight under all the assignments $f_1, f_2, \ldots, f_t$, as follows. Since f_1 maps the nonempty subsets of S to the range $[1, NB]$, there exists a family A_1 of nonempty subsets of S, all of which get the same weight under f_1 and such that $|A_1| \geq \frac{2^N-1}{NB}$. Similarly, there exists a family $A_2 \subseteq A_1$ with $|A_2| \geq \frac{2^N-1}{(NB)^2}$, such that $f_i(X_1) = f_i(X_2)$, for all X_1, X_2 in A_2, for $i = 1, 2$. Repeating this, we end up with a set A_t of nonempty subsets of S with $|A_t| \geq \frac{2^N-1}{(NB)^t}$, such that the minimum weight element of A_t is not unique under any of $f_1, f_2, \ldots, f_t$; since $\frac{2^N-1}{(NB)^t} \geq 2$ for $t = o(\frac{N}{\log N})$, our claim holds. Thus, any scheme for Generalized Isolation requires $\Omega(\log Z + \log N)$ random bits. Theorem 2.1 also gives similar lower bounds when the elements are assigned superpolynomial weights. In fact, we can prove the same lower bound when the weight assignments map subsets of S to an arbitrary linearly ordered set of size B since the above proofs use no property of the integers other than their total ordering. Also the functions need not assign the weight of the subsets to be the sum of the elements in the subset. Notice that this lower bounds only holds when the isolating scheme assigns weights to the ground set elements. In the next chapter we shall show that any abstract scheme such as the Isolating Lemma must use $O(\log Z + \log n)$ random bits thus giving a stronger lower bound.

2.2 Applications to Matching Problems

The Generalized Isolating Lemma immediately yields randomness-efficient algorithms for several problems related to matching. In this section, we only consider matchings in bipartite graphs in order to simplify the exposition; we sketch the ideas for matchings in general graphs and omit the details which can be inferred from the standard approaches [LP86, MVV87]. The failure probabilities of all our algorithms can be made inverse polynomial with only a constant blowup in the time and randomness complexities and a polynomial blowup in the number of processors using two point sampling techniques of Chor & Goldreich [CG89]. Another important feature of parallel algorithms which we shall not pay particular attention to is that of processor efficiency: our algorithms tend to use more processors than previously known algorithms. The justification for this attitude is that our isolating lemma is a general tool to bound the randomness complexity of a problem in terms of the number of its solutions and the main import of our approach is that we can unify many results which are obtained by dissimilar techniques. However, due to the large polynomial weights used by our lemma, algorithms based directly on the lemma usually suffer a processor penalty.

Notation In the following subsections we shall denote a bipartite graph G by $G = (U, V, E)$, where $U = \{u_1, u_2, \ldots, u_{n/2}\}$, $V = \{v_1, v_2, \ldots, v_{n/2}\}$, and $|E| = m$. The determinant of a matrix A is denoted by $det(A)$. The number of perfect matchings in G is, as usual, denoted by $|\mathrm{Mat}(G)|$.

2.2.1 Algorithms for matching

To apply the Isolating Lemma to the perfect matching problem, we let the ground set be the set of edges of the input graph G and let the family of subsets be the set of perfect matchings of G, following the approach of [MVV87].By our generalization of the Isolating Lemma, given an upper bound Z on $|\mathrm{Mat}(G)|$, we can assign polynomially bounded weights to the edges of G using $O(\log Z + \log n)$ random bits such that there is a unique perfect matching of minimum weight, with good probability. We construct a matrix $M[1..n/2, 1..n/2]$ with

$$M[i,j] = \begin{cases} 2^{w_{ij}} & \text{if } (u_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

where w_{ij} is the weight assigned to edge (u_i, v_j). A perfect matching in G can now be found, if one exists, by inverting M via the NC^2 algorithm of Csanky [Csa76], as shown in [MVV87]. Note further that the additive $O(\log n)$ factor in the randomness complexity can be absorbed by a bigger processor penalty. For any graph $G = (V, E)$, $|\mathrm{Mat}(G)| \leq \prod_{v \in V} degree(v)$, which is at most $(2m/n)^n$ since $\sum_{v \in V} degree(v) = 2m$. In the worst case, by letting $Z = (2m/n)^n$ we obtain a randomness complexity of $O(n \log(m/n))$ which improves the previous bound of $O(m \log n)$ [MVV87]. Thus, we get

Theorem 2.2 *There is an RNC^2 algorithm to find a perfect matching in a graph G (if one exists) which uses $O(\log Z + \log n)$ random bits, given an upper bound Z on $|\mathrm{Mat}(G)|$. In the worst case, this algorithm uses $O(n \log(m/n))$ random bits.*

A careful analysis of the proof of our isolating scheme when applied to the worst case where $Z = (2m/n)^n$ and $m = \Theta(n^2)$, shows that the following algorithm also achieves the same randomness complexity: the edges of each vertex $u \in U$ are assigned weights *pairwise independently* in the range $[1, m^7]$. In fact, the starting point of this entire work was this observation.

Perfect matching is a special case of the *Exact Matching Problem*: given a graph G with edges colored red and blue arbitrarily and an integer k, an exact matching is a perfect matching of G with *exactly* k red edges (Papadimitriou & Yannakakis [PY82]). Even though the problem of finding an exact matching is not known to be solvable sequentially in polynomial time, the Isolating Lemma has been used to solve it in RNC^2 [MVV87]! By observing that isolation is needed only among the exact matchings, we can link the randomness complexity of this problem to a bound X on the number of exact matchings and *not* on the number of perfect matchings. Suppose we assign a nonnegative integral weight w_{ij} to every edge $(u_i, v_j) \in E$. Let y be an indeterminate; construct a symbolic matrix $M_y[1..n/2, 1..n/2]$ such that

$$M[i,j] = \begin{cases} y \cdot w_{i,j} & \text{if } (u_i, v_j) \in E \text{ and is colored red,} \\ w_{ij} & \text{if } (u_i, v_j) \in E \text{ and is colored blue,} \\ 0 & \text{otherwise} \end{cases}$$

Then, it is easy to see that the contribution to the coefficient of y^k in $det(M_y)$ comes precisely from the exact matchings in G and that if there is a unique minimum weight exact matching M^*, then its weight equals the highest power of 2, say p^*, which divides this coefficient of y^k. Furthermore, any edge can be tested for membership in M^* by increasing its weight by 1 and testing if the new value of p^* is higher than its old value or not. Hence, this problem can be solved by using our Generalized Isolating Lemma over the ground set E and the (unknown) family of exact matchings of G. Since the determinant of a matrix in one variable can be computed in NC^2 (Borodin, Cook & Pippenger [BCP83]), we get

Theorem 2.3 *Given an upper bound X on the number of exact matchings in a graph G, an exact matching (if any) in G can be found in RNC^2 using $O(\log X)$ random bits.*

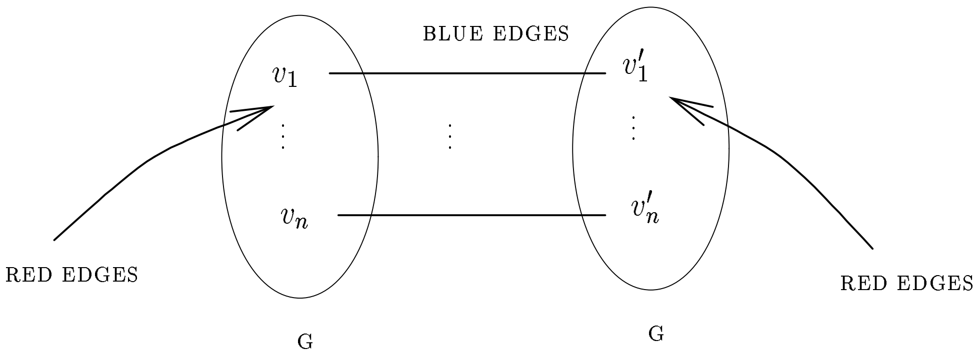

Figure 2.2: Reduction from maximum matching to exact matching which prevents blowup in number of solutions.

The Isolating Lemma has also been used to find the matching of largest cardinality in bipartite graphs by a reduction to the perfect matching problem. However the NC reductions that have been used earlier [MVV87,LP86] do not suffice to get randomness–efficient solutions since they cause a blow up in the number of solutions. Instead, we use the following reduction from maximum matching to exact matching: Given a graph $G = (V, E)$ we first construct 2 copies of G and the edges are colored red. Each vertex v in the first copy of G is joined to vertex v in the second copy by a blue edge. to yield the resulting graph, G' as shown in Figure. 2.2. If M is a matching in the original graph of size m then there is a a perfect matching in G' with $2m$ blue edges: Take the edges of M in the 2 copies of the graph G and match each vertex v unmatched in the matching M with its corresponding vertex v' in the other copy of G. Conversely, any perfect matching with with l blue edges in the graph G' defines a matching in the original graph G of size at least $\lceil \frac{l}{2} \rceil$. Thus it follows that if m is the size of the largest matching

---

Input: Graph $G = (V, E)$ and bound Z on number of maximum matchings

Construct graph G' by making 2 copies of G and coloring edges as in Figure 2.2

For each value of m from 1 to $\frac{|V|}{2}$ *in parallel* do

Run Exact Matching algorithm to find a matching in G' with $2m$ blue edges assuming a bound of Z^2 exact matchings.

Output the largest sized matching of G obtained in all parallel runs.

Figure 2.3: Randomness–Efficient algorithm for maximum matching

in G then there is an exact matching with $2m$ blue edges in G' but no exact matching with more than $2m$ blue edges. Also, since the only way to get an exact matching with $2m$ blue edges in G' is to choose a maximum matching of G in both the copies, if Z is an upper bound on the number of maximum matchings in G, G' has at most Z^2 exact matchings with $2m$ blue edges. Thus this new reduction to exact matching does not cause more than a polynomial blowup in the number of solutions. Using this we get a randomness–efficient algorithm for perfect matching which is described in Figure 2.3. The correctness of the algorithm follows from the correctness of the reduction and the algorithm for exact matching.

Theorem 2.4 *There exists an RNC^2 algorithm for maximum matching which uses $O(\log Z + \log n)$ random bits, where Z is a known bound on the number of maximum matchings in G.*

2.2.2 Graphs with a polynomial number of perfect matchings

Grigoriev & Karpinski [GK87] consider the problem of finding all the perfect matchings in graphs with a polynomial bound on the number of perfect matchings. Using techniques from algebra, they give an NC^2 algorithm for this problem if the number of matchings is bounded by $\log^{0.5-\epsilon} n$, and an NC^3 algorithm when the bound is n^c for constant c. Since their algorithm to extract a matching is limited by a procedure to detect if a matching exists they suggest that $\log^{0.5-\epsilon} n$ may be a limiting upper bound on the number of matchings to find all (or even one) of the perfect matchings in NC^2 via parallel algebra. We present a NC^2 algorithm for the detection problem which provides enough information to find all matchings in NC^2 even when the number of matchings is polynomially bounded. In addition to the improvement in running time, our algorithm is much simpler than the algorithm of Grigoriev & Karpinski.

As before, we let S and $\mathcal{F}$ be E and $\mathrm{Mat}(G)$ respectively. Running steps 1 and 2 of the generalized Isolating scheme, we use $O(\log n)$ random bits to assign each edge a polynomially bounded weight such that *all* the perfect matchings of G have *distinct* weights with probability at least half. Let $w_{i,j}$ be the weight assigned to edge (u_i, v_j). Construct a matrix A with

$$A[i,j] = \begin{cases} 2^{2\times w_{ij}} & \text{if } (u_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Since every matching has a distinct weight, $det(A)$ gives information about *every* perfect matching as shown below.

Assume that Step 2 of our generalization of the Isolating Lemma succeeds, *i.e.*, all the perfect matchings indeed get distinct weights. If we denote the matrix obtained from

M by removing its ith row and jth column by M_{ij}, then by definition

$$(M^{-1})[j,i] = \frac{(-1)^{i+j} det(M_{ij})}{det(M)}.$$

Now, if there is a perfect matching of weight k containing the edge (u_i, v_j), then $det(M_{ij})$ is of the form

$$\pm 2^{2\times(k-w_{ij})} + \sum_{\ell=0}^{k-1} a_\ell 2^{2\times(\ell-w_{ij})} + \sum_{\ell \geq k+1} a_\ell 2^{2\times(\ell-w_{ij})},$$

where each a_ℓ is in $\{-1, 0, 1\}$. Since

$$|\sum_{\ell=0}^{k-1} a_\ell 2^{2\times(\ell-w_{ij})}| < 2^{2\times(k-w_{ij})-1},$$

this is of the form

$$\pm 2^{2\times(k-w_{ij})} \pm n_0 \times 2^{2(k-w_{ij}+1)} \pm x \times 2^{2(k-w_{ij})-1},$$

where n_0 is a nonnegative integer and $0 \leq x < 1$. Similarly, if there is no perfect matching of weight k containing the edge (u_i, v_j), then $det(M_{ij})$ is of the form

$$\pm n_0 \times 2^{2(k-w_{ij}+1)} \pm x \times 2^{2(k-w_{ij})-1}.$$

Hence, there is a perfect matching of weight k containing the edge (u_i, v_j) if and only if

$$|(M^{-1})[j,i] \cdot det(M) \cdot 2^{2w_{ij}}| \pmod{2^{2(k-1)}}$$

is of the form $4n_1 \pm y$, where n_1 is an integer and $y = 1$ or 2. If there is no matching of weight k then $y = 0$ or 3. Since k is polynomially bounded we can extract all matchings. Also, since we use $O(\log Z + \log n) = O(\log n)$ random bits, we can derandomize this by searching over all points in the sample space in parallel to get

Theorem 2.5 *There is an NC^2 algorithm to detect if a perfect matching exists and to find all perfect matchings in a graph G, when given that $|\mathrm{Mat}(G)| \leq n^c$ for some constant c.*

We can obtain the same result for exact matching and maximum matching problems given a polynomial bound on the number of exact matchings and maximum matchings respectively.

Grigoriev & Karpinski also give a Las Vegas RNC^3 algorithm for the problem of testing if a graph has at most n^c perfect matchings, which uses expected $O(n^{2c+8.5}\log n)$ random bits. We can, using the algorithm of Theorem 2.5 and by extending ideas of [KVV85,Kar86], improve on the running time and substantially on the number of random bits used, as follows: Karloff presents an NC^2 algorithm which calls a perfect matching algorithm (oracle) on $n+1$ graphs (each with $n-1$ or n vertices and at most m edges) in parallel, and which, assuming that these calls worked correctly, will announce *correctly* if G has a perfect matching. or not; in any case, if it ever says that $|\mathrm{Mat}(G)| = 0$, then the number of perfect matchings is indeed zero. Actually, Karloff's result is more general – it works for maximum matchings. Note that in conjunction with any Monte Carlo RNC^2 algorithm to find a perfect matching (in particular, ours) this yields a Las Vegas RNC^2 algorithm to decide if G has a perfect matching or not, and if so, to produce one. We first make the error probability of our perfect matching algorithm at most $\frac{1}{2(n+1)}$ (as mentioned in Section 2.1, this can be done in RNC^2) and using only $O(n\log(m/n))$ random bits then invoke Karloff's algorithm, which will make calls in parallel to our algorithm, using the *same sequence of random bits* for each call; the probability that at least one of these calls produced a wrong result is at most $\frac{n+1}{2(n+1)} = 1/2$. Hence, we can *correctly* decide if $|\mathrm{Mat}(G)| > 0$ or not, using an expected $O(n\log(m/n))$ many random bits. The above probability can be made any inverse polynomial with a processor penalty using the techniques of two–point sampling[CG89] If we conclude that $|\mathrm{Mat}(G)| > 0$, then we assume that $|\mathrm{Mat}(G)| \leq n^c + 1$, and run the algorithm of Section 2.2.2 to generate all the perfect matchings of G. Notice, however, that even if there are exponentially many matchings in the graph the procedure in Section 2.2.2 might yield a few matchings in the graph. Thus we have to verify that the graph has no other matchings. Kozen, Vazirani and Vazirani[KVV85] give a simple NC^2 algorithm to test given a perfect matching whether there are other perfect matchings in the graph. We extend this result to obtain an NC^2 procedure for the following problem: Given a polynomial number of perfect matchings of the graph, are there any other perfect matchings?

Lemma 2.2 *Given a bipartite graph G and a nonempty collection of at most polynomial number of perfect matchings of G, there exists an NC^2 procedure to test if the graph contains other perfect matchings.*

PROOF. Let $G = (U, V, E)$ be the given graph and let $\mathcal{M}$ be the given non–empty, polynomial sized collection of perfect matchings of G. We first choose a matching $M_1 \in \mathcal{M}$ and renumber the vertices so that M_1 corresponds to the identity permutation *i.e.* M_1 is the perfect matching $(u_1, v_1), \ldots, (u_n, v_n)$. We then construct the modified adjacency matrix M' of G as follows:

$$M'[i,j] = \begin{cases} x & \text{if edge } (u_i, v_j) \text{ is in } M_1 \\ y & \text{if edge } (u_i, v_j) \text{ is in any of the matchings in } \mathcal{M} - M_1 \\ 1 & \text{if edge } (u_i, v_j) \text{ is not in any of the matchings of } \mathcal{M} \\ 0 & \text{otherwise} \end{cases}$$

For each of the matchings in $\mathcal{M}$ we compute the multinomial term obtained by the above weight assignments, *i.e.*, the product of the sign of the permutation corresponding to the matching and the weights of the edges according to the above assignments. For instance, the term corresponding to M_1 will be x_n. Let $p_{\mathcal{M}}(x, y)$ denote the multinomial obtained by summing the terms for all the matchings in $\mathcal{M}$. Let $q(x, y)$ be the symbolic determinant of the above matrix. Then we claim that the graph G has no matchings other than the ones in $\mathcal{M}$ if and only if $p_{\mathcal{M}}(x, y) = q(x, y)$. Clearly, if no other matchings exist then $p_{\mathcal{M}}(x, y) = q(x, y)$. Assume that there is at least one other matching. Then for each matching M which is not in $\mathcal{M}$ the symmetric difference of M and M_1 consists of cycles. Let k be the length of the smallest such cycle. The sign of all the matchings with weight $x^{n-k}y^r$ for some $r \geq 0$ is $(-1)^{k-1}$ since the symmetric difference of such matchings and M_1 must be a single k length cycle, by definition of k. Thus if there are other perfect matchings in G then the two polynomials will differ in the coefficient of x^{n-k}. Since the determinant of M' can be evaluated in NC^2 by the result of[BCP83] the lemma follows. □

The Las Vegas algorithm will check using the above procedure if there are other matchings in the graph and accept if there are no other matchings otherwise it reports a failure. If G has at most n^c perfect matchings, then at most n^c perfect matchings will

Input: Graph $G = (V, E)$.

for $i := 0$ **to** $\lceil \log_2 \log_2((2m/n)^n) \rceil$ **do**

Let $Z = 2^{2^i}$ and run Perfect Matching algorithm of Section 2.2.1 to find a perfect matching using Z as bound on number of matchings.

exit if a perfect matching is found.

If no perfect matching is found then $|\mathrm{Mat}(G)| = 0$.

Figure 2.4: Randomness–efficient perfect matching algorithm with unknown bound

be generated and we will pass the test for other matchings and so the algorithm accepts correctly with high probability.

Theorem 2.6 *There exists a Las Vegas RNC^2 algorithm to test if a given graph G has* $|\mathrm{Mat}(G)| \leq n^c$ *whose expected randomness complexity is* $O(n \log(m/n))$.

Since the lemma can be proved for proved for general graphs as well the above theorem holds also for general graphs.

2.2.3 Randomness-Efficient algorithms with no information on the number of matchings

The algorithms of Theorem 2.2 can be used to obtain randomness–efficient algorithms for perfect matching even when no upper bound on the number of perfect matchings is given, by using the worst case upper bound of $Z = (2m/n)^n$. However, the algorithm shown in Figure 2.4 has a much better randomness complexity. If there is at least one perfect matching in the graph G, then this algorithm will find a perfect matching with good probability when $2^{2^i} \geq |\mathrm{Mat}(G)|$. If M is the number of perfect matchings then since the algorithm definitely succeeds when $2^{2^{i-1}} \leq M \leq 2^{2^i}$ and in this stage the algorithm uses $O(2^i)$ random bits. Therefore the randomness–complexity of the algorithm is $O(\log M)$ and the algorithm runs in time $O(\log^2 n \log\log M)$ with high probability. The correctness of the algorithm can be easily shown and hence we obtain the following theorem.

Theorem 2.7 *Let M=$|\mathrm{Mat}(G)|$. There is an RNC algorithm to find a perfect matching in G with no given upper bound on M, which uses $O(\log M)$ random bits and runs in time $O(\log^2 n \log\log M) = O(\log^3 n)$ with high probability, if $M > 0$; in the* worst case, *it uses $O(n\log(m/n))$ random bits and $O(\log^3 n)$ time.*

2.3 Other applications

One of the advantages of designing algorithms with techniques like the Isolating Lemma is that they are very abstract and powerful tools and can be used in a variety of situations. The lemma has been used in many situations where we are required to make one solution from a possibly exponential sized collection stand out. This abstraction because of its general nature has several applications. Thus, in all these situations our generalized Isolating Lemma leads to randomness–efficient solutions to various problems, which we describe in this section. We start with applications which are essentially generalizations of matching such as problems on linear matroids and T–joins.

2.3.1 Improved parallel randomness complexity for problems on matroids

The Isolating Lemma has been used to obtain RNC algorithms for basic problems on linearly representable matroids such as matroid intersection and matching (Narayanan, Saran & Vazirani [NSV92]). These problems are generalizations of bipartite and general graph matching.

Definition A set system $M = (S, \mathcal{F})$ is a *matroid* if:

- $\phi \in \mathcal{F}$,
- if $A \subseteq B \subseteq S$ and $B \in \mathcal{F}$, then $A \in \mathcal{F}$, and
- if $A_1 \in \mathcal{F}$ and $A_2 \in \mathcal{F}$, then $\exists x \in A_2 - A_1 : A_1 \cup \{x\} \in \mathcal{F}$.

Every element of $\mathcal{F}$ is called an *independent set.* A basic consequence of the matroid axioms is that every maximal independent set is also a maximum independent set, and the cardinality of any maximum independent set of M is also called the *rank* of M. A matroid M of rank r over a ground set S of cardinality n is *linearly representable* over a field $\mathbf{F}$ if there exists a matrix $C \in \mathbf{F}^{r \times n}$ whose columns are indexed by the elements of S, such that a subset of S is independent in M iff the corresponding set of columns of C are linearly independent over $\mathbf{F}$. M is *linearly represented* if it is presented as the matrix B. Henceforth, the field $\mathbf{F}$ is assumed to be the field of rationals, $\mathbf{Q}$. All the results of [NSV92] and of this section hold *only* for linearly represented matroids. For a good introduction to the very well studied theory of matroids see for example Welsh[Wel76] and Lovasz–Plummer[LP86]. Camerini, Galbliati and Maffioli[CGM92] have also dealt with the problem of randomized algorithms for the matroid problem, although with the focus of obtaining random pseudopolynomial sequential algorithms for generalizations of these problems.

The matroid intersection problem is to find a maximum cardinality set which is independent in both of two given matroids M_1 and M_2, each of rank r and over the same ground set S of cardinality n. In this case, we follow the algorithm as presented in [NSV92] and for completeness we describe their algorithm with the necessary modifications in Figure 2.5. We actually use the following modification of the Generalized Isolating Lemma.

Lemma 2.3 *Let $(S,\ \mathcal{F})$ be any set system and let Z be a* given *upper bound on the size of the unknown family $\mathcal{F}$. Assume the elements of S are given initial weights $u_1, \ldots, u_n$ bounded in magnitude by U. There is a simple scheme which uses $O(\log Z + \log N + \log U)$ random bits to assign integer weights $w_1, \ldots, w_n$ to the elements of S in the range $[0, N^7 + N^7 \log U)$, such that if the weight of element x_i is defined to be $w_i + u_i$, then with probability at least $1/4$, there is a* unique *minimum weight set in $\mathcal{F}$.*

The proof of this modification of the Lemma is a straightforward extension of the old proof. In fact, the correctness of the algorithm for maximum matching follows from the correctness of this modification of the Isolating Lemma. By carefully counting we can observe that the family that the isolating lemma is applied is at most $I \times (\binom{r}{r-h})^2 \times (r-h)!$ where h is the size of the largest set in $M_1 \cap M_2$ and I is the number of sets of size h in $M_1 \cap M_2$. Since this can be bounded by $(nr^2)^h \times r! \leq (nr^2)^r \times r^r \leq n^{4r}$, the modified generalized isolation process can be used to reduce the randomness complexity from the previous bound of $O((n + r^2) \log n)$ [NSV92] to $O(r \log n)$, improving by a factor of at least $\Omega(\sqrt{n})$ in all cases, and by more in general. If $h \geq r - O(\log_r n)$ and there is a

Input: Matrices A and B, elements of $\mathbf{Q}^{n\times r}$

/* A and B are linear representations of matroids M_1 and M_2 of rank at most r */

Let $e_1, \ldots, e_r$ be the unit vectors in $\mathbf{Q}^r$

Construct matrices $C, D \in \mathbf{Q}^{r\times r^2}$ such that for each pair i, j

there is some k such that the k^{th} columns of C and D are e_i and e_j.

Let $A' = [A, C]$ and $B' = [B, D]$

Use modified Isolating lemma with ground set as the columns of A' and initial weights of the columns $u_j = 0$ for $1 \leq n$ and $u_j = r(n+r)^8$ for $j > n$. Let the final weights of elements be $w_i, \ldots, w_{n+r^2}$.

/* Family of subsets is all choices of r indices such that the corresponding columns of both A' and B' are independent. */

Multiply i^{th} column of A' by x^{w_i}.

Compute $\det(A'(B')^T)$. The coefficient of the minimum power of x corresponds to the isolated maximum cardinality intersection and can be extracted in parallel.

Figure 2.5: Randomness–efficient algorithm for matroid matching

polynomial (in n) bound on I then since $I(\binom{r}{r-h})^2 \times (r-h)! = n^{O(1)}$, we get the first NC algorithms for the matroid intersection problem. In the case of bipartite matching which is a special case of matroid intersection, $h = r$, and a polynomial bound on I corresponds exactly to a polynomial bound on the number of matchings in the input graph. Thus, this gives one way to generalize the results of [GK87] to matroids.

Theorem 2.8 *Matroid intersection for linearly represented matroids of rank r can be solved in RNC^2 using $O(r \log n)$ random bits. If the cardinality of the maximum intersection is given to be at least $r - O(\log_r n)$ and if the number of maximum cardinality intersections is given to be bounded by a given polynomial of n, then it can be solved in NC^2.*

For linearly represented vectors, the matroid matching problem is the following: given m pairs of vectors over $\mathbf{Q}^{2n}$, pick the largest number of pairs so that the vectors picked are linearly independent. For this problem, the RNC^2 algorithm of ([NSV92], Theorem 4.3) uses the Isolating Lemma on a set family over a ground set of $m + \binom{2n}{2}$ elements and at most

$$\binom{m+\binom{2n}{2}}{n} \leq (m + 2n^2)^n$$

subsets. By replacing their isolation step by the modification of the generalized isolating lemma we can improve the randomness complexity from $O((m+n^2)\log(m+n^2))$ [NSV92] to $O(n \log(m + n^2))$. In fact, we can count the size of the size of the family, to which the Isolating Lemma is applied, more carefully to bound the randomness complexity of the algorithm to be $O(\log(I \times \binom{2n}{2n-2h} \times (2n - 2h)!))$ where h is the maximum number of pairs that can be picked so that the union is linearly independent and I is the number of ways these h pairs can be picked. As in the case of matroid intersection we can obtain deterministic algorithms if I is polynomial and if $h \geq n - \log_n m$. Tiwari, using sparse interpolation techniques, gives NC algorithms for this problem when the number of full dimensional solutions (*i.e.*, when $h = n$) is given to be polynomially bounded in n and m [Tiw87]. Thus, our new NC algorithms are applicable to a larger range of problems than the result of Tiwari and our randomized algorithm is a generalization. Since the algorithm follows directly from the algorithm of [NSV92] and the above modification of the Generalized Isolating Lemma we omit the details of the algorithm.

Theorem 2.9 *Matroid matching can be solved in RNC^2 using $O(n \log(m+n^2))$ random bits. If $h \geq n - \log_n m$, is the size of the largest number of pairs whose union is linearly independent and there is a polynomial bound on the number of ways such a set of h pairs can be picked, matroid matching can be solved in NC^2.*

2.3.2 Algorithms for other generalizations of matching

The concept of a T–join in a graph is another way to generalize the idea of perfect matchings. Given a graph $G = (V, E)$ and a set $T \subseteq V$ of even size, a set $F \subseteq E$ is called a T–join if the degree of vertices in T is odd and vertices not in T have even degree, where degrees are counted in the subgraph (V, F). A minimum T–join is one of minimum cardinality. The minimum T–join problem is to find a minimum T–join given input $G = (V, E)$ and $T \subseteq V$. The perfect matching problem is a special case of the T–join problem with $T = V$.

This problem can be solved by a standard reduction to the minimum weight perfect matching problem in graphs with weights given in unary[Wei91,LP86]. For this problem using the modification of the Generalized Isolating Lemma which we described in the previous subsection we can derive randomness–efficient algorithms for the general case and deterministic algorithms for special cases when the number of solutions is small. Since the algorithms can be obtained by directly plugging in the modification to our isolation process we omit the details and refer the interested reader to the previous algorithms[MVV87,LP86,Wei91].

2.3.3 Randomized reductions from SAT to USAT

An important result in complexity theory is the random reduction, due to Valiant & Vazirani, from languages in NP to USAT the language of uniquely satisfiable Boolean formulae [VV86]. This reduction is at the core of several fundamental results in complexity theory, most notably in the results of Toda on the unexpected power of counting classes [Tod91]. The Isolating Lemma has been used to derive an alternative random reduction in [MVV87]. We apply the Isolating Lemma to get a slightly different reduction and here, our generalization yields better randomness complexity. We describe this in the next chapter where we consider this reduction in detail and such several alternate reductions using our generalization of the Isolating Lemma and other techniques from hashing theory. We will also generalize the lower bound for isolation to hold for many generalizations of the weight assignment problem.

2.4 Discussion and open problems

The main contribution of this chapter is our Generalized Isolating Lemma. For various problems solvable via randomness in general [MVV87,NSV92,VV86] and deterministically when the number of solutions is small [KVV85,GK87,CH90], this lemma bounds their randomness complexity in terms of the number of solutions. Our results which span the spectrum of the number of solutions, imply, and sometimes improve the previously known results at the extremes. On the other hand, our lower bound for isolation implies that attempts to solve these problems deterministically based on isolation must also exploit their structure, rather than view them merely as abstract unknown collections of sets.

A obvious open problem suggested by the Generalized Isolating Lemma is to reduce the magnitude of the weights assigned by it to the elements of the ground set; currently, they can be as high as N^7. In the context of matching, this will lead to more processor–efficient RNC algorithms. The lower bound for isolation that we prove holds only for schemes which work oblivious of the family of solutions that it is tring to isolate one solution from. An interesting question that arises here is: can we use some knowledge of the structure of the solution space that we are trying to isolate one solution from, in order to reduce the randomness complexity of isolation or even perhaps obtain deterministic algorithms? In case of the perfect matching problem it would be interesting to see if we can obtain deterministic algorithms for special classes of graphs with this framework.

Chapter 3

Efficient Random reductions

Random reductions were first introduced by Adleman and Manders[AM77] in order to give evidence of the hardness of certain number theoretic problems which were not known to be complete for NP under deterministic reductions. For an excellent source on the properties of random reductions we refer the reader to [Roh94]. A very important random reduction in structural complexity due to Valiant and Vazirani, is the reduction of problems in the class NP to the language, USAT, of uniquely satisfiable Boolean formulas[VV86]. This result showed that although USAT may not be complete for NP under deterministic reductions, it is as hard as the problems in NP. This reduction has played a central role in the area of structural complexity theory as many results use this random reduction. One of most important uses of this reduction has been in the result of Toda[Tod91]. The reduction of Valiant and Vazirani actually showed that languages in NP randomly reduce to a language in Parity Polynomial time($\oplus$P) and Toda's breakthrough work generalized this to show that languages in the entire Polynomial Hierarchy randomly reduce to $\oplus$P.

In this chapter we will focus on efficient random reductions from languages in the class NP to the language of uniquely satisfiable formulas and to languages in the class Parity Polynomial time. We will first provide an alternative reduction of languages in NP to $\oplus$P which is very simple and intuitively appealing. We then study random reductions from the point of view of randomness efficiency, *i.e.*, to use as few random bits as possible. It can easily be shown that the original reduction of Valiant and Vazirani requires $O(n^2)$ random bits in the worst case. A variant of this reduction was obtained by Mulmuley, Vazirani and Vazirani using the Isolating Lemma and the randomness complexity of this reduction is $O(n \log n)$. We will show that the probabilistic tool that was developed in the previous chapter gives a more efficient random reduction from NP to $\oplus$P. We also observe that several techniques in hashing and coding theory yield random reductions which use $O(n)$ random bits in the worst case. We will then extend some of these reductions to obtain reductions which use $O(\log Z + \log n)$ random bits where Z is an upper bound on the number of satisfying assignments of the input formula, which gives the worst case complexity of $O(n)$ random bits. Moreover, just as we did for the case of the other applications of the Generalized Isolating Lemma from the previous chapter, we will consider this reduction for special cases. In the case of formulas with polynomial bound on the number of satisfying assignments, such as in the case of languages in FewP, which are accepted by machines which categorically accept on only polynomially many paths, the reductions can be derandomized to get new proofs of the result that FewP $\subseteq \oplus$P[CH90] and other results.

We also study lower bounds on the number of random bits required for such reductions from NP to the class $\oplus$P. In the area of structural complexity, one way to deal with problems on the structure of classes of languages is to consider the same questions under different computation models. A clean way to do so is to take the standard Turing machine models and augment them with a new primitive of being able to decide membership in a fixed oracle language A for free. This provides a new set of complexity classes relative to the new oracle language A. Then, it is common practice to study the usual questions in this relativized world. For an introduction to this area of structural complexity we refer the reader to [BGS75].

Although, all languages in NP randomly reduce to languages in $\oplus$P, it is well known that there is no relativizing deterministic reduction[BG82]. However, when the number of accepting paths is categorically bounded by a fixed polynomial there is a deterministic reduction, *i.e.*, FewP $\subseteq \oplus$P and this result relativizes. A natural question that arises is: how much can we increase the categorical bound on the number of accepting paths and still obtain a relativizing deterministic reduction to $\oplus$P? We show that if the categorical bound is Z, then $\Theta(\log Z - c \log n)$ (for any constant c) is a tight bound on the number of random bits used by relativizing randomized reductions to $\oplus$P. Thus, when Z is polynomial, deterministic reductions are possible; for larger Z, no relativizing deterministic reductions are possible. Also, our random reductions, which are relativizing, are optimal in the use the of random bits. These lower bounds rule out several approaches to obtaining deterministic algorithms for the perfect matching, using techniques which are generalizations of the isolation process discussed in the previous chapter.

3.1 Preliminaries

In this section, we define some complexity classes and other preliminaries required for the exposition in the rest of the chapter and summarize their basic properties. We also describe the Valiant–Vazirani random reduction and other known reductions which are relevant to the rest of the exposition. We assume that the reader is familiar with basic models of computation and computational complexity classes. In particular we assume familiarity with complexity classes such as NP and machine models such as oracle Turing machines.

Definition 1 [PZ83] *A language L is in the class* $\oplus$P *if there exists a nondeterministic polynomial time Turing machine N such that a string x is in L if and only if $N(x)$ accepts on an odd number of paths.*

$\oplus$P is one of the simplest language classes one can define whose membership is defined by a simple predicate on the number of accepting paths of a nondeterministic polynomial time Turing machine. The class was originally defined by Papadimitriou and Zachos[PZ83] who established many closure properties of $\oplus$P. In the subsequent sections we will require that $\mathrm{P}^{\oplus \mathrm{P}} = \oplus\mathrm{P}$[1].

Definition 2 *The language* USAT *is the set of Boolean formulas with exactly one satisfying assignment.*

The random reduction of Valiant and Vazirani from languages in NP to USAT comes from the following powerful lemma.

[1] $\mathrm{P}^{\oplus \mathrm{P}}$ is the class of languages that can be accepted by polynomial time oracle Turing machines with the languages in $\oplus$P as oracles.

Lemma 3.1 *Let* $S_0 \subseteq Z_2^n$ *such that* $S_0 \neq \emptyset$. *Choose vectors* $w_1, \ldots, w_n$ *uniformly at random from* Z_2^n. *Define the sets* $S_1, \ldots, S_n$ *recursively as*

$$S_{i+1} = \{ \ v \in S_i \mid v \cdot w_{i+1} \ = \ 0 \ \}$$

where $x \cdot y$ *denotes the inner product of vectors* x *and* y. *Then, with probability at least* $\frac{1}{4}$, *one of the sets* $S_0, \ldots, S_n$ *has* exactly one element.

To obtain a random reduction, given a formula F, let S_0 be the set of satisfying assignments of F and define a formula F_i whose satisfying assignments are exactly S_i. Then one of the formulas is uniquely satisfiable iff F is satisfiable. Also, if F is not satisfiable then none of the formulas is satisfiable. Thus the above lemma also yields a random reduction from NP to $\oplus$P.

A different technique to isolate a single element from such an implicitly given set was devised by Mulmuley, Vazirani and Vazirani[MVV87]. Their reduction arises from their Isolating Lemma described in the previous chapter.

Lemma 3.2 [MVV87] *Let* S *be a ground set of elements* $\{x_1, \ldots, x_n\}$ *and* $\mathcal{F}$ *be a family of subsets of* S. *If the elements of* S *are assigned weights independently and uniformly at random from the range [1,2n] then the minimum weight subset in* $\mathcal{F}$ *is unique with probability* $\frac{1}{2}$.

Given a formula $F(x_1, \ldots, x_n)$, the above lemma states that if we assign weights to the x_i's, and let the weight of a satisfying assignment be the sum of the weights of the variables which are set to true in that assignment, then we isolate a unique minimum weight satisfying assignment.

In this chapter we will derive many alternate reductions to USAT and $\oplus$P, some of which are simple and more intuitive than the above reductions and others which are more randomness efficient. One of the reductions follows from our generalization of the Isolating Lemma in the previous chapter. The basic tools that the other reductions use will be techniques from hashing and coding theory. A class of hash functions which will be used often is the class of universal hash functions. These are typically useful when no assumptions can be made on the distribution of the domain to be hashed.

Definition 3 [WC79] *Let H be a set of functions $h : \{0,1\}^n \rightarrow \{0,1\}^n$. H is called a d-universal family of hash functions if for any $x_1 \neq x_2 \ldots \neq x_d$ and elements $y_1, \ldots, y_d$ in $\{0,1\}^n$ we have that*

$$\mathrm{Prob}_{h \in H}[h(x_1) = y_1 \text{ and } h(x_2) = y_2 \ldots \text{ and } h(x_d) = y_d] = (\frac{1}{2^n})^d = \frac{1}{2^{nd}}$$

The collection, H, of degree $d-1$ polynomials with coefficients in GF$[2^n]$ is a d-universal family of hash functions for all $d \leq n$.

3.2 A Simple reduction from NP to $\oplus$P

In this section we describe a new technique to isolate an odd number of witnesses (satisfying assignments) which directly yields a reduction from NP to $\oplus$P. Consider the following simplistic approach: Treat satisfying assignments as n-bit strings, choose a random n-bit cut off point r and discard all assignments lexicographically greater than r, hoping that an odd number of them remain. This naive approach will not work if the initial set is not well distributed, e.g., if the initial even-sized set consists of a few witnesses which are clustered together, then the probability of retaining an odd number of witnesses will be extremely low.

To overcome this obvious drawback we adopt a simple strategy: Given a set of witnesses, S, consider the set $h(S)$, where h is a hash function. If h is a good hash function then the elements of $h(S)$ will be well spread out over $\{0,1\}^n$. Then, for a randomly chosen vector r, the size of the set $S' = \{x \mid h(x) < r\}$ will be odd with good probability. Since the hash functions must work for all sets of vectors the universal hash functions defined by Carter and Wegman [WC79] are an obvious choice. The following theorem establishes the new isolating technique. Choose $d = c\frac{n}{\log n}$ for some even constant c.

Theorem 1 *Let S be any collection of n bit vectors. Choose h randomly from a d-universal family, H, of hash functions and r a random point in $\{0,1\}^n$. Then the set $S_1 = \{x \mid x \in h(S) \text{ and } x < r\}$ has an odd number of elements, with probability at least $\frac{1}{n^3}$.*

This yields a reduction from NP to $\oplus$P, as shown in Fig. 3.1, which has a success probability $\frac{1}{n^3}$. The reduction also generalizes to other levels of the PH by the techniques of the original proof of Toda[Tod91].

Given a set S of N vectors, we would ideally like the hash functions to map them such that the gap between successive elements is close to $\frac{2^n}{N}$ so that a randomly chosen point would have an odd number of elements to its left with high probability. Even if the sum of the odd gaps, *i.e.*, the gaps between even numbered elements are their predecessors in the lexicographic order, is an inverse polynomial fraction of the total, we get adequate success probability. We show in Lemma 3.3 that given a set S, for a large fraction of

INPUT: Boolean formula F of length n.

- Randomly choose h from a family of d-universal hash functions and a point $r \in \{0,1\}^n$.
- Construct an NP machine N which on input F,h, and r guesses a and verifies that a satisfies F and $h(a) < r$.
- Use Cook's reduction to obtain a formula $F'(F,h,r)$ which is satisfiable iff N accepts on input F,h and r.

Figure 3.1: A simple reduction from NP to $\oplus$P.

hash functions, h, in every d consecutive elements of $h(S)$ there is at least one odd gap of size at least $\frac{2^n}{N\times d^2}$ or one even gap of size at least $\frac{2d2^n}{N}$. In Theorem 1 we show that this ensures that the success probability for the entire reduction is at least $\frac{1}{d^3}$.

Before we proceed to the details of the proof we define some notation: The even gaps are the gaps between odd numbered elements and their predecessors (if any) in lexicographic order. Similarly odd gaps are gaps between even numbered elements and their predecessors.

Definition 4 *A hash function, h, is* bad *for $S = \{a_1, \ldots, a_N\}$ if there exists $S' \subseteq S$ of size d such that every odd gap of $h(S')$ is of size less than $g_o = \frac{2^n}{N\times d^2}$ and every even gap of $h(S')$ is of size less than $g_e = \frac{2d2^n}{N}$. h is* good *if it is not* bad *for S.*

It is easy to see why, for a fixed d element set S', few of the functions from a d-universal family are bad: if a function is bad then every element must be mapped close to its predecessor. When h is chosen at random, $h(S')$ is a set of d independent random variables and hence the probability that all of them are close to their predecessors is very low.

Lemma 3.3 *Let H be a d-universal family of hash functions and S be a set of N vectors in $\{0,1\}^n$. Then with probability at least $\frac{1}{2}$, a randomly chosen hash function is good for S.*

Proof: Let $S' \subseteq S$ with $|S'| = d$ and let $b_1, \ldots, b_d$ be an arbitrary permutation of the elements of S'. We now count the number of hash functions h which are bad for S' and which map the elements in the same order. There are 2^n choices to map the first element b_1, but once $h(b_1)$ is fixed, $h(b_2)$ must be within distance g_o of $h(b_1)$ and $h(b_3)$ must within distance g_e of $h(b_2)$ and so on. In general there are g_o places to map an even numbered element and g_e places to map an odd numbered element. For each choice of d positions to map the elements of S' to, there is exactly one hash function which maps the elements to those positions. Thus the probability that a randomly chosen hash function maps elements in this fixed order and is bad for S' is at most $p_0 = 2^n \times g_o^k \times g_e^{k-1} \times 2^{-nd}$ where $k = \frac{d}{2}$.

The probability a randomly chosen function is bad for any one of the $d!$ permutations of S' is at most $d! \times p_0$. Since there are $\binom{N}{d}$ subsets of S of size d the probability that a randomly chosen hash function is bad for S is at most $\binom{N}{d} \times (d)! \times p_0$ which is at most $N^d \times p_0$. Substituting the values of g_o and g_e we can choose $d = \frac{8n}{\log(n)}$ and the probability that a randomly chosen hash function is bad is at most $\frac{1}{2}$. $\square$

Proof: (of Theorem 1) Given a set $S = \{a_1, \ldots, a_N\}$, choose a hash function h, that is good for S. Let $h(S) = \{b_1, \ldots, b_N\}$ with $b_1 \leq b_2 \ldots \leq b_N$. Since h is good for S, for every d element subset $h(S)$ there is at least one odd gap of size $g_o = \frac{2^n}{N \times d^2}$ or at least one even gap of size $g_e = \frac{2d \times 2^n}{N}$. Thus in each of the d element subsets $\{b_1, \ldots, b_d\}$, $\{b_{d+1}, \ldots, b_{2d}\} \ldots$, either the sum of the odd gaps is at least g_0 or the sum of the even gaps is at least g_e. Since $g_e = \frac{2d \times 2^n}{N}$, only half the $\frac{N}{d}$ subsets can have the sum of even gaps greater than g_e, otherwise their sum would exceed 2^n. So, the sum of the odd gaps in at least $\frac{N}{2d}$ subsets is more than $g_o = \frac{2^n}{N \times d^2}$. Since d is even, any point in the odd gaps of these subsets will have an odd number of elements to its left. Thus if h is good, a randomly chosen point will, with probability $\frac{1}{2 \times d^3}$, have an odd number of elements to its left. By Lemma 3.3 at least half the hash functions are good for S. □

The analysis in Theorem 1 is loose since we discard hash functions which are bad on even one d sized subset. A tighter analysis may show that a smaller value of d is sufficient. Using techniques of probability amplification without wasting random bits[Nis90] and the fact that ⊕P is closed under majority reductions we can reduce the error probability of the reduction to inverse exponential with only $O(n^2)$ random bits.

3.3 Randomness efficiency of reductions

In this section we consider the randomness efficiency of reductions from NP to USAT and ⊕P. We first construct an example on which the Valiant–Vazirani reduction and a slight generalization will be forced to use $\Theta(n^2)$ random bits. Then we show that several techniques from coding theory and hashing yield reductions whose worst complexity is $O(n)$ random bits. We also give an alternate random reduction using our generalization of the Isolating Lemma. All these reductions have better randomness complexity than the $O(n \log n)$ complexity using the isolating technique of Mulmuley, Vazirani and Vazirani. We also extend several of these techniques to obtain reductions whose randomness complexity is $O(\log Z + \log n)$ where Z is an upper bound on the number of satisfying assignments of the input formula. The class FewP [All86] consists of languages in NP that can be accepted by nondeterministic machines which *categorically* accept on at most a fixed polynomial number of paths. For languages in FewP our random reductions can be derandomized, since they use only $O(\log n)$ random bits and we can try out all possible values of the random bits. We show that our efficient random reductions yield new proofs of the result that FewP$\subseteq \oplus$P[CH90] and other results.

3.3.1 A lower bound on the complexity of the Valiant–Vazirani reduction

We consider the following obvious generalization of the unique element isolating technique of Valiant and Vazirani.

Definition 5 *Let $S_0 \subseteq \{0,1\}^n$ such that $S_0 \neq \emptyset$ and let b be a randomly chosen vector in $\{0,1\}^n$. Choose vectors $w_1, \ldots, w_n$ uniformly at random from Z_2^n. Treat elements of S_0 as vectors in Z_2^n. Define the sets $S_1, \ldots, S_n$ recursively as*

$$S_{i+1} = \{\ v \in S_i \mid v \cdot w_{i+1} \ = \ b_{i+1}\ \}$$

where $x \cdot y$ denotes the inner product of vectors x and y.

Note that the original reduction is a special case when the vector $b = 0^n$. It can be shown that for most sets S, after the first step S_1 is odd sized with high probability. Thus it may appear that a few steps of the reduction is sufficient to produce a set of odd size.

However there are easily constructed initial sets for which the reduction requires $\Omega(n)$ steps and that the first set of odd size is of size one.

Example 1 *Let $S_0 = \{0,1\}^n$. Then for any choice of b, if z_i is a fixed element of S_i then the set $T_i = \{y \mid y + z_i \in S_i\}$ is a vector space of dimension at least $n - i$.*

So for this initial set the reduction will require $\Omega(n)$ steps and the first set of odd size will be of size one. However, the formula whose satisfying assignments correspond to this initial set is easy to recognize. An instance that is not trivial is the following:

Example 2 *Let $S_0 = \{t + x \mid t \in T\}$ where T is any k-dimensional subspace of Z_2^n and x is a vector in Z_2^n. Call such a set a* bad set of order k. *Then it can be shown that S_i is a bad set of order at least $k - i$.*

A bad set of order k can be shown to require at least k steps of the reduction before an odd sized set results, irrespective of the choice of the vectors $w_1, \ldots, w_n$ and b. With this initial set we can construct hard instances of Boolean formulas for which the reduction requires $\Omega(n^2)$ random bits.

3.3.2 Efficient reductions

As shown above the Valiant–Vazirani random reduction requires $\Omega(n^2)$ random bits in the worst case. By definition the random reduction of Mulmuley, Vazirani and Vazirani requires $\Theta(n \log n)$ random bits. In this section we observe that several techniques from hashing and coding theory will yield random reductions which are more randomness–efficient. The worst case complexity of all the reductions in this section is $O(n)$ random bits. We extend all these reductions to obtain reduction whose complexity is $O(\log Z + \log n)$ random bits where Z is a known upper bound on the number of satisfying assignments. As mentioned before, these can be derandomized to obtain new proofs of results in structural complexity theory.

A reduction with two–universal functions

We will now sketch a reduction based on a simple property of two-universal hash functions. The basic idea of this reduction is that we hash the set of satisfying assignments, if any, into a range which is of size at most a constant times the number of satisfying assignments. We do this while ensuring that with high probability a constant fraction of values in the range will have exactly a single satisfying assignment mapping on to them. Thus, if we choose a random value, r, in the range and retain only the satisfying assignments that map to r, then with high probability we will isolate a single satisfying assignment. The details are formalized by the following lemma which appears in many guises in hashing theory.

Lemma 3.4 *Let $x_1, \ldots, x_m$ be distinct elements in $\{0,1\}^n$. Let $\mathcal{H}$ be a family of 2–universal hash functions from $\{0,1\}^n$ to $[1, 8m]$. Then*

$$\mathrm{Prob}_{h\in\mathcal{H},y}[\text{ there exists a single } i \text{ such that } \; h(x_i) = y] \geq \frac{1}{64}$$

where y is chosen uniformly from $[1, 8m]$.

Proof: We give a short and elementary proof. Let $h \in \mathcal{H}$ be chosen randomly and let Y_j be the random variable that is 1 if there exists some x_i which maps to j and 0 otherwise, where $1 \leq j \leq cm$. We wish to estimate the expected value of Y_j. Using the

Inclusion–Exclusion principle, for a fixed j, we get

$$\mathrm{Prob}_h[\exists i\ h(x_i) = j\ ] \ \geq\ \sum_i \mathrm{Prob}_h[h(x_i) = j\ ] - \sum_{i_1 \neq i_2} \mathrm{Prob}_h[h(x_{i_1}) = j\ \&\ h(x_{i_2}) = j\ ]$$

Since $\mathcal{H}$ is a family of 2-universal hash function, by definition, for a fixed i, the probability that x_{i_1} maps to j is $\frac{1}{8m}$ and for fixed i_1, i_2 the probability that both x_{i_1} and x_{i_2} map to j is $\frac{1}{64m^2}$. Thus

$$\mathrm{Prob}_h[\ (\exists i)\ h(x_i) = j\ ] \ \geq\ m \times \frac{1}{8m} - \binom{m}{2} \times \frac{1}{64m^2} = \frac{1}{8} - \frac{1}{128}$$

By linearity the expectation of the random variable $S = \sum Y_j$ is at least $m(1 - \frac{1}{16})$. Note that S is a random variable whose maximum value is m. Since the expectation is close to the maximum value, the probability that $S \geq \frac{3m}{4}$ is at least $\frac{1}{2}$ for otherwise the expected value of S is at most $\frac{3m}{4} \times \frac{1}{2} + m \times \frac{1}{2} = \frac{7m}{8}$ which is a contradiction. Thus, with high probability $S \geq \frac{3m}{4}$. Since m objects are mapped into at least $\frac{3m}{4}$ distinct values this implies that at least $\frac{m}{4}$ of the values in the range $[1, 8m]$ have exactly one x_i mapping to them. Thus the lemma follows. □

Given a formula F, if we know the number of assignments that satisfy F then the above lemma can be used to get a reduction. To obtain a reduction which runs in polynomial time, we first guess an integer k in the range $[0, n-1]$. Assume that the formula has between 2^k and 2^{k+1} and use 2-universal hash functions to hash them to a space of size 2^{k+4}. This yields a reduction as shown in Fig. 3.2 which succeeds with probability at least $\frac{1}{128 \times n}$.

We also wish to point that several other properties of 2–universal hash functions also yield reductions from NP to USAT which use $O(n)$ random bits.

INPUT: Boolean formula F of length n.

- Randomly choose k from the range $[0, n-1]$, h from a family of 2–universal hash functions from $\{0,1\}^n$ to $\{0,1\}^{k+4}$ and a point $r \in \{0,1\}^{k+4}$.
- Construct an NP machine N which on input F, h and r guesses a and verifies that a satisfies F and $h(a) = r$.
- Use Cook's reduction to obtain a formula $F'(F,\ h,\ r)$ which is satisfiable iff N accepts on input F, h and r.

Figure 3.2: Efficient reduction from NP to USAT using 2–universal hash functions.

Reductions with bounds on number of satisfying assignments

In this section we consider reductions when there are known bounds on the number of satisfying assignments of the input formula. As an example, languages in the class FewP are accepted by nondeterministic machines which categorically accept on at most a fixed polynomial number of paths. A natural language which is in the class FewP is primality testing[FK92].

A reduction using the Generalized Isolating Lemma

Given a formula $F(x_1, x_2, \ldots, x_n)$ as input, to use the isolating lemma we let the ground set S be the set of variables $\{x_1, x_2, \ldots, x_n\}$ and let the family $\mathcal{F}$ be the satisfying assignments of F, where a satisfying assignment is represented by the subset of variables that are true in it. Assign weights $w_1, \ldots, w_n$ to the ground set elements as prescribed by the generalized isolating lemma and choose a random element y in the range [1,n^8]. Consider the following reduction:

On input $F(x_1,\ldots,x_n), w_1,\ldots,w_n, y$ we construct an NP machine M which works as follows: Guess an assignment for the variables $x_1,\ldots,x_n$. If the assignment satisfies the formula F and the weight of the assignment under the weight assignment $w_1,\ldots,w_n$ is equal to y then we accept. Notice that since the weight assignment scheme succeeds with probability $\frac{1}{2}$, if F is satisfiable then with probability $\frac{1}{2}$ there is one minimum weight satisfying assignment. Since all the assignments have a weight in the range $[1,n^8]$ and y is chosen uniformly at random from $[1,n^8]$ we have

$$F \text{ is satsfiable} \Longrightarrow \text{Prob}[\ M \text{ accepts on exactly one path }] \geq \frac{1}{2\times n^8}$$

If F is not satisfiable then the machine M clearly accepts on no paths. Given this machine M we can construct using the Cook–Levin reduction a boolean formula F' which has exactly the same number of satisfying assignments as M. So the random reduction first assigns weights to the variables, chooses a random weight y and then constructs the formula F' described above. By the description above it is easy to see that

$$F \text{ is satisfiable } \Longrightarrow \text{Prob}[\ F' \text{ is uniquely satisfiable }] \geq \frac{1}{n^8}$$

$$F \text{ is not satisfiable } \Longrightarrow \text{Prob}[\ F' \text{ is not satisfiable }] = 1$$

This reduction requires $O(\log Z + \log n)$ random bits, where Z an upper bound on the number of satisfying assignments of F. In the worst case this randomness complexity is $O(n)$, which improves on the $O(n\log n)$ bound of [MVV87], the $O(n^2)$ bound of [VV86] and matches the result of Tarui [Tar91]. Thus we obtain:

Theorem 3.1 *There is a randomized reduction from SAT to USAT which uses* $O(\log Z + \log n)$ *random bits, where* n *and* Z *are the number of variables and an upper bound on the number of satisfying assignments respectively.*

Extending the reduction with 2–universal hash functions

We will extend the reductions of the previous section to derive reductions that use $O(\log Z + \log n)$ random bits where Z is an upper bound on the number of satisfying assignments of the input formula. Note that in those reductions the randomness complexity is $O(n)$ due to the complexity of the 2–universal hash functions. For the correct guess of the range in which m, the number of satisfying assignments, lies, hashing is required from a domain of size $O(m)$ to a range of size $O(m)$. However since the actual satisfying assignments are unknown we are forced to hash from a domain of size $O(2^n)$ to $O(m)$. In order to rectify this we will first reduce the size of the domain to be hashed to $O(m^2)$ and then carry out the reduction. To do this we use the known upper bound on the number of satisfying assignments, Z, to assign an $O(m)$ bit ID to each of the n bit assignments such that with high probability all the satisfying assignments get distinct ID's. Then we can use the 2–universal hash function as before on the smaller ID's which are distinct to carry out the reduction. The following lemma shows one way in which this can be done.

Lemma 3.5 *Let* $x_1 \ldots, x_m$ *be* n *bits numbers. If* l *is a randomly chosen integer in the range* $[1, n^2 \times m^4]$ *and* $y_i = x_i \bmod l$ *then*

$$\text{Prob[the } y_i\text{'s are distinct]} \geq \frac{1}{2}.$$

Proof: A short proof can be obtained using a well known number theoretic lemma. The version of the lemma we use is due to Tompa[Tom91](see also [Thr93]). If S is any subset of $\{1,\ldots,k^2\}$ of size at least $\frac{k}{2}$ then the least common multiple of the elements of S is at least 2^k[Tom91]. Denote by p the product $\Pi_{i<j}(x_i - x_j)$. Then the magnitude of p is at most $2^{n\times\binom{m}{2}}$ since each term of the product is at most 2^n. By the number theoretic fact, at most half the integers from the range $[1, n^2 \times m^4]$ can divide p since otherwise the least common multiple of the divisors, which exceeds p, must divide p. Thus with probability at least a half the choices of l will not divide p. Thus for these choices of l, the y_j's will all be distinct. $\square$

Thus the modified reduction which given an upper bound, Z, on the number of satisfying assignments will first choose a random number in the range $[1, n^2Z^4]$ which is used to reduce n bit ID's to $O(\log Z + \log n)$ bit ID's and then carry out the other steps of the reduction choosing the hash functions whose domain is $\{0,1\}^r$ and the value of k is restricted to be in the range $[0, \lceil \log Z \rceil - 1]]$. Thus the randomness complexity of the entire reduction will be $O(\log Z + \log n)$.

As before there are many ways to accomplish the same reduction in the size of the domain, for example other hashing techniques[DGMP92]. We will describe another technique based on sampling from robust encodings of binary strings using some ideas in [ALMS92]. Though this technique is no more efficient from the above method, the method is novel.

Theorem 3.2 *Let $x_1, \ldots, x_m$ be n bit integers. There exists a simple probabilistic function $f : [1, m] \rightarrow [1, 2^{O(\lceil \log m \rceil)}]$ using $O(\log m)$ random bits such that*

$$\text{Prob}[f(i) \;\neq\; f(j) \text{ for all } i, j] \geq 1 - \frac{1}{m^{O(1)}}$$

PROOF. We choose a prime p of magnitude roughly n^{10}, say the first prime of magnitude greater than n^{10}, which we can choose by testing exhaustively for primality. Given an n bit number x we treat the number as defining a polynomial f_x of degree n and the n bits are the values of f_x on n fixed members of the finite field Z_p^*, say $1, 2, \ldots, n$. The degree n polynomial f_x can be then determined by standard interpolation techniques. The probabilistic function given a sequence of random bits works as follows: Given i we construct f_{x_i} and form an $O(\lceil \log m \rceil)$ bit number by using the given random bits to choose $\frac{\lceil \log m \rceil}{\log n}$ values from the finite field Z_p^* and output the binary representations of the values of f_x evaluated at these randomly chosen points. The lemma follows from the fact that for different i and j the degree n polynomials f_{x_i} and f_{x_j} are different and hence can not evaluate to the same value at more than n different points of the finite field. Since the field is chosen to have about n^{10} points they evaluate differently at a randomly chosen point. □

We can derandomize the above reductions for languages in FewP, to get a polynomial number of formulas such that F is satisfiable if and only if one of these formulas is uniquely satisfiable. Also, if F is unsatisfiable then none of these formulas is satisfiable. Given a set of such formulas it is easy to test in $P^{\oplus \mathrm{P}}$ if at least one of them is uniquely satisfiable by querying the $\oplus$P oracle and accepting if at least one of the formulas has an odd number of satisfying assignments. When F is unsatisfiable none of the set of formulas is satisfiable, so no formula has an odd number of formulas. Thus we can obtain a new simple proof of the result that FewP $\subseteq \oplus$P.[CH90].

We also use the above reduction to get a proof that FewP $\subseteq \mathrm{C}_=\mathrm{P}$.

Definition 2 *A language L is in the class* $\mathrm{C}_=\mathrm{P}$ *if there is a nondeterministic polynomial* *time machine M and a polynomial time computable function f such that x is in L if and only if M on input x accepts on exactly $f(x)$ paths.*

We can use $C_{=}P$ to test if one of a polynomial number of formulas has exactly one satisfying assignments as follows: Let $F_1, \ldots, F_t$ be formulas with $s_1, \ldots, s_t$ satisfying assignments respectively. Suppose we could construct a formula F' with $(s_1-1)\cdots(s_t-1)$ satisfying assignments. Then F' would have exactly 0 satisfying assignments iff one of the F_js has exactly one satisfying assignment. However doing this directly involves constructing formulas with one less satisfying assignment than the F_js which is highly intractable[OH91]. We do this slightly indirectly as follows: Each term in the product $(s_1-1)\cdots(s_t-1)$ is of the form $\pm s_{i_1}s_{i_2}\cdots s_{i_r}$ where $r \leq t$. Using standard techniques, we can construct a formula which has $s_{i_1}s_{i_2}\cdots s_{i_r}$ satisfying assignments. Now if the coefficient of this term is $+1$ then we are done. However if the coefficient of this term is -1 then we construct a formula with $2^{p(n)} - s_{i_1}s_{i_2}\cdots s_{i_r}$ satisfying assignments, where $p(n)$ is a large polynomial such that $2^{p(n)}$ is greater than all terms. Such a formula is easy to construct from a formula F'' which has $s_{i_1}s_{i_2}\cdots s_{i_r}$ satisfying assignments: consider the NP machine which has $2^{p(n)}$ total paths and which accepts only on paths which do not give satisfying assignments for F''. Thus for each negative term we add exactly $2^{p(n)}$ satisfying assignments. Lastly note it now easy to define a formula G with exactly $(s_1-1)\cdots(s_t-1) + 2^{p(n)} \times 2^{t-1}$ satisfying assignments since there are 2^{t-1} terms with a coefficient -1. Now G has exactly $2^{p(n)} \times 2^{t-1}$ satisfying assignments iff one of the original formulas has exactly one satisfying assignments. Thus these efficient reductions also give new proofs of FewP $\subseteq$ $C_{=}$P .

We emphasize here that though there are many proofs of the above two results the main advantage of our proofs is that they follow directly by bounding the randomness complexity of the general randomized reduction.

3.4 Lower bounds

In the previous sections we have given randomized reductions from NP to $\oplus$P which uses $\mathrm{O}(\log Z + \log n)$ random bits. This leads to the question of whether a reduction with a better randomness complexity can be found. The kind of reductions we focus on in this section is the one–sided error ($\leq_m^{\mathrm{rp}}$) random reductions.

Definition 6 *A language $A \leq_m^{\mathrm{rp}}$ reduces to a language B if there is a probabilistic transducer M such that*

$$x \in A \Rightarrow \text{Prob.}[M(x) \in B] \geq 1/2$$
$$x \notin A \Rightarrow \text{Prob.}[M(x) \in B] = 0$$

It can be verified that all the reductions that we have seen so far are one–sided error random reductions. Also, we focus on relativizing reductions since all known reductions from NP to USAT and $\oplus$P relativize. By an old result, it is known that there are no relativizing deterministic reductions from NP to USAT[BG82]. However, if we only consider languages in NP accepted by machines which categorically accept on only a polynomial number of paths then there is a deterministic reduction, *i.e.*, FewP $\subseteq \oplus$P. A natural question to ask is what the randomness complexity is when we raise this categorical bound. In this section we characterize this exactly by showing that a language is accepted by a machine which always accepts on at most $Z(n)$ paths then any relativizing reductions from this language to $\oplus$P must use $\mathrm{O}(\log Z(n))$ random bits.

We emphasize that such a lower bound holds only for relativizing reductions. However, this result can be used to rule out certain approaches to the problems we have considered. The main application of the Isolating Lemma of Mulmuley, Vazirani and Vazirani[MVV87] is to derive RNC algorithms for the maximum matching problem. However, due to its general nature it yields random reductions from NP to USAT. In the previous chapter a lower bound of $O(\log Z + \log n)$ bits is given for the specific scheme used by Mulmuley *et al.* The following construction will establish the same lower bound for *any* general probabilistic scheme similar to the Isolating Lemma, which is general enough to yield reductions from NP to $\oplus$P.

Specifically we construct an oracle A such that the language

$$L(A) = \{0^n \mid \exists y \ |y| = n \text{ and } y \in A\}$$

is accepted categorically by a nondeterministic oracle machine on at most $Z(n)$ paths *with this oracle.* We will diagonalize to ensure no probabilistic transducer which uses less than $(\log(Z(n)) - c\log n)$ random bit, for all constants c, can reduce $L(A)$ to the standard complete language for $\oplus\mathrm{P}(A)$. This establishes that any relativizing reduction from NP to $\oplus$P must use at least $\log(Z(n))$ bits. Our construction is an extension of the technique of [Bei91].

Theorem 2 *Let $Z(n)$ be a function of the form $2^{k(n)} - 1$. Then there exists an oracle A such that $L(A)$ is accepted by a nondeterministic machine which accepts on at most $Z(n)$ paths with oracle A on all inputs and such that no reduction using less than $\log(Z(n)) - c\log n$ random bits (for all constants $c > 0$) can be a one sided error random reduction from $L(A)$ to the standard complete language for $\oplus\mathrm{P}(A)$ with even nonzero probability.*

Proof: We denote the standard complete language for $\oplus$P(A) by $L_0(A)$. Though we have not yet constructed the oracle A as yet, we can take the universal language $\{(N_i, x, 1^k) \mid N_i^A(x)$ accepts on an odd number of paths in k steps$\}$ as the complete language $L_0(A)$ for the rest of the proof. Let M_0 be a nondeterministic oracle machine such that $x \in L_0(A)$ iff $M_0^A(x)$ accepts on an odd number of paths and let $q(n)$ be the running time of M_0 on inputs of length n. Let R_1, $R_2, \ldots$ be an enumeration of probabilistic transducers using less than $\log Z(n) - c \log n$ random bits for all constants $c > 0$, with the running time of R_i bounded by $p_i(n) = n^i + i$. We construct an oracle A such that all potential reductions R_i, from $L(A)$ to $L_0(A)$, do not work correctly with even a nonzero probability.

The oracle A is constructed in stages and in stage i we diagonalize against the transducer R_i. Let $n_0 = 0$ and $A_0 = \emptyset$. In stage i we extend the oracle A by an extension E_i , *i.e.*, $A_{i+1} = A_i \cup E_i$. Finally we set $A = \bigcup A_i$.

Description of Stage i: Choose $n > n_{i-1}$ such that $2^n > q(p_i(n))$. In further stages we only add strings of length greater than 2^n and this condition ensures that later stages do not interfere with the oracle answers for the construction in this stage. In stage i we will also maintain $E_i \subseteq \{0,1\}^n$.

First set $E_i = \emptyset$. Let $y_1, \ldots, y_t$, where by assumption $Z(n) > tn^c$ for all $c > 0$, be the different outputs of the probabilistic transducer R_i on input 0^n over all its random paths. For each j, run M_0 on input y_j with oracle A_{i-1}. If for some j, $M_0^{A_{i-1}}(y_j)$ accepts on an odd number of paths then we fix $E_i = \emptyset$, set $n_i = 2^n$ and move to the next stage. Notice that since $E_i = \emptyset$ and we never add strings of length n to A in later stages, the string 0^n is not in $L(A)$. Also since we $2^n > p_i(n)$ we freeze all the oracle responses to M_0 on input 0^n and therefore $R_i(0^n)$ is in $L_0(A)$ with nonzero probability and hence we have diagonalized against R_i to ensure that it is not a $\leq_m^{\mathrm{rp}}$ reduction from $L(A)$ to $L_0(A)$.

Assume that $M_0^{A_{i-1}}(y_j)$ accepts on an even number of paths for all j. Call an extension $E_i \subseteq \{0,1\}^n$ valid if $0 < |E_i| \leq Z(n)$. Also, for technical reasons, we shall only consider an extension as valid if it does not contain the string $x_0 = 1^n$. We now show, by contradiction, that there exists some valid extension E_i such that $M_0^{A_{i-1} \cup E_i}(y_j)$ accepts on an even number of paths for all j. We then extend A by such an E_i at this stage. Since $E_i \neq \emptyset$ we have $0^n \in L(A)$ but by construction we will have $y_j \notin L_0(A)$ for all j. Thus the potential random reduction R_i from $L(A)$ to $L_0(A)$ fails to work with probability 1 on the input 0^n since at least one of its outputs will be accepted by M_0^A. Note that by definition of a valid extension, we only put in at most $Z(n)$ strings of length n. This ensures that the obvious nondeterministic machine to accept $L(A)$ will accept on at most $Z(n)$ paths on all inputs. Thus for this oracle A, the language $L(A)$ is in FewP(A).

Claim 3.1 *There exists a valid extension E_i such that $M_0^{A_{i-1} \cup E_i}(y_j)$ accepts on an even number of paths for all j.*

Proof: Assume, if possible, that for all valid extensions E_k there exists some y_{j_k} such that $M_0^{A_{i-1} \cup E_k}(y_{j_k})$ accepts on an odd number of paths. Fix some valid extension E_k. Let $P_j(k)$ be the set of accepting paths of $M_0^{A_{i-1} \cup E_k}$ on the input y_j. Define the set

$$\mathrm{Labels}(E_k) = \{\#\} \bigcup [(P_1(k) \cup \{\#\}) \times (P_2(k) \cup \{\#\}) \cup \cdots \cup (P_t(k) \cup \{\#\})].$$

The idea behind this definition is that $|\mathrm{Labels}(E_k)|$ is odd iff $|P_j(k)|$ is odd for some j. By assumption, for each E_k, there exists some j_k such that $|P_{j_k}(k)|$ is odd. Thus $|\mathrm{Labels}(E_k)|$ is odd for each valid extension E_k. By definition the number of valid extensions is $\binom{2^n-1}{1} + \cdots + \binom{2^n-1}{Z(n)}$ since we always omit the string x_0 from the oracle. Using a theorem due to Edouard Lucas which appears as exercise in [Knu82] we can deduce that $\binom{2^n-1}{r}$ is odd for all r. Since $Z(n)$ is odd we can deduce that there are an odd number of valid extensions. Thus $\sum_{\text{valid } E_k} |\text{Labels}(E_k)|$ is odd. We will show by counting this sum in a different way that the sum is even which is a contradiction and thus our initial assumption is false.

For any valid extension E_k there are 2 trivial labels, namely # and (#,...,#). We will argue that all nontrivial labels $\ell = (l_1, \ldots, l_t)$ occurs in Labels(E_k) for an even number of valid extensions E_k and these two facts imply that the sum $\sum_{\text{valid } E_k} |\text{Labels}(E_k)|$ is even. Consider a nontrivial label $\ell = (l_1, \ldots, l_t)$ which occurs in Labels(E_k) for some valid extension E_k. By definition, there is some tuple l_j which corresponds to an accepting path of M_0. For each such tuple which corresponds to an accepting path identify which strings are queried by M_0 along this path and let S_j be the set of strings in the oracle of length n which are in the oracle and R_j be the set of strings of length n which are not in the oracle. If tuple l_j is the label # then let $S_j = R_j = \emptyset$. Define $S_\ell = \cup S_j$ and $R_\ell = \cup R_j$. Notice that ℓ will occur in Labels(E'_k) for any valid extension E'_k which contains all the strings in S and none of the strings in R. The maximum size of the sets S_ℓ and R_ℓ is at most $tq(p_i(n))$ and by choice $Z(n) - |S_\ell| > tq(p_i(n))$. If $S_\ell \neq \emptyset$ then the number of valid extensions E'_k for which ℓ occurs in Labels(E'_k) is $\binom{2^n-1-s-r}{0} + \binom{2^n-1-s-r}{1} + \cdots + \binom{2^n-1-s-r}{Z(n)-s}$ where $s = |S_\ell|$ and $r = |R_\ell|$. By a careful application of Lucas' Theorem noting that $Z(n) - |S_\ell| > tq(p_i(n))$ and $Z(n) = 2^{k(n)} - 1$ we can derive that $\binom{2^n-1-s-r}{x}$ is odd for an even number of x between 0 and $Z(n) - s$. Thus ℓ appears in an Labels(E'_k) for an even number of valid extensions E'_k if S_ℓ is non empty.

Now consider the case that S_ℓ is empty. Here, to count the valid extensions E'_k for which ℓ appears in Labels(E'_k) we must choose to add at least one string in E'_k since otherwise the extension is not valid. Thus the number of valid extensions E'_k for which ℓ occurs in Labels(E'_k) is $\binom{2^n-1-r}{1} + \cdots + \binom{2^n-1-r}{Z(n)}$ which is odd. We will argue that the number of such labels is even. Notice that tuples of ℓ which are accepting paths also remain accepting paths when M_0 is run with the oracle A_{i-1} , *i.e.*, with an empty extension at this stage. By assumption $M_0^{A_{i-1}}(y_j)$ accepts on an even number of paths for each j. Thus to count the number of nontrivial labels with $S_\ell = \emptyset$, which can appear in Labels(E'_k) for some valid extension E'_k, we just consider paths which accept on the empty extension at this stage as possible choices for the nontrivial entries of the label ℓ. By assumption, for the empty extension for each j, $M_0(y_j)$ accepts on an even number of paths. Therefore the number of nontrivial labels which can occur in Labels(E'_k) for some valid extension E'_k is even. Putting all the steps together we can derive that $\sum_{\text{valid } E_k} |\text{Labels}(E_k)|$ to be even, which is a contradiction.

To visualize the above argument, think of a table whose rows are indexed by valid extensions of the oracle and columns by labels. Place a mark on the entry (k, ℓ) if the label ℓ occurs in Labels(E_k). Now if the statement of the claim is false then each row has an odd number of marks and by counting we showed that there are an odd number of rows. However, counting the number of marks in the columns yields an even sum which is a contradiction. For a column corresponding to a nontrivial label ℓ with $S_\ell \neq \emptyset$ we argued that there are an even number of marks in that column. The remaining argument showed that the number of columns corresponding to nontrivial labels with $S_\ell = \emptyset$ is even. Thus if we count the number of marks in each column the sum is even. This contradicts the fact that sum along the rows is odd. □

3.4.1 Implications of the Lower bound

The lower bound is shown by constructing an oracle to diagonalize against all reductions and thus it holds only relativizing reductions. This style of lower bound proof once popular in the world of structural complexity in now no longer in vogue due mainly to the recent results on interactive proofs[ALMS92]. However here is a concrete instantiation of this result which rules out some approaches to finding deterministic algorithms for the perfect matching problem. Consider the isolation lemma from the previous chapter where we assigned weights to the ground set elements to ensure that the minimum weight subset was unique. A relaxation of this which could be used for problems like detecting the existence of perfect matching is to assign weights such that there is an *odd* number of subsets of *some* weight. This seems to be a huge relaxation since the obvious entropy style reasoning for lower bounds does not hold. However, since this would also yield a relativizing random reduction from NP to $\oplus$P the above result implies a lower bound of $\log Z(n)$ random bits on such a scheme. In fact, we arrived at the lower bound when we discovered the lower bound for the above scheme.

3.5 Discussion

We have given several alternatives to the Valiant–Vazirani random reduction from NP to USAT and $\oplus$P. Some of these reductions are much simpler and elegant than the previously known reductions while others are more efficient in the use of randomness.

We also show a lower bound on the randomness complexity of any relativizing random reduction from NP to $\oplus$P. The open question that remains is if there are non relativizing random reductions and on a bigger scale if there are deterministic reductions.

Chapter 4

Efficient Parallel Algorithms via Approximations of Probability Distributions[1]

One of the big motivations for designing probabilistic algorithms is that for a large class of problems these probabilistic algorithms can be directly converted to deterministic algorithms by the process of derandomization. There have been a number of significant advances in the development of general tools to derive efficient *deterministic* algorithms from their randomized counterparts over the last few years. The best-known deterministic solutions for several classical problems such as testing undirected graph connectivity within limited space bounds, result from this [NSW92].

The three major known approaches to derandomization are the techniques of limited

[1]The results presented in this chapter are joint work with Pankaj Rohatgi and Aravind Srinivasan and a preliminary version appears in the Proceedings of the 26^{th} Annual Symposium on the Theory of Computing.

independence ([KW85,Lub86,ABI86]), the method of conditional probabilities ([Rag88, Spe87]), and small–bias probability spaces ([NN90], and also [AMN90,AGHP92,ABN$^+$92]).

Definition 3 *Random variables $X_1, X_2, \ldots, X_n$ are k–wise independent if for any set $I \subseteq \{1, \ldots, n\}$ of at most k indices and for any choice $v_1, \ldots, v_n$ we have*

$$\mathrm{Prob}[\wedge_{i \in I}(X_i = v_i)] = \prod_{i \in I} \mathrm{Prob}[X_i = v_i]$$

Efficient constructions of sample spaces $S \subseteq \{0,1\}^n$, of size smaller than 2^n, such that the distribution induced on $\{0,1\}^n$, by sampling uniformly from S, is k–wise independent are given in [Lub86,Jof74,ABI86].

Definition 4 *If $S \subseteq \{0,1\}^n$ is a small–bias space with parameter ϵ then if $\vec{X} = (X_1, \ldots, X_n)$ is sampled uniformly from S, then for all sets $I \subseteq \{1, \ldots, n$ and for all bits $b_1, \ldots, b_{|I|} \in \{0,1\}$*

$$|\mathrm{Prob}(\wedge_{i \in I} X_i = b_i) - 2^{-|I|}| \leq \epsilon.$$

Again the main motivation behind this is that such sample spaces can be constructed with size much smaller than 2^n. This is very useful in derandomization, since a large class of randomized algorithms are robust to such small changes in the probabilities.

The method of conditional probabilities can be informally described as follows: Suppose the algorithm chooses n random bits $\vec{X} = (X_1, \ldots, X_n)$ and outputs a function $f(\vec{X})$. Typically, by analysis we can show that $E(f(\vec{X}))$ is large and, by definition this means that there exists $\vec{X_0}$ such that $f(\vec{X_0}) \geq E(f(\vec{X}))$. The goal is to find this point $\vec{X_0}$. The method of conditional probabilities is to find this point by setting the components of $\vec{X_0}$ one bit at a time. We do this so that at every stage the expectation of $f(\vec{X})$ conditioned on fixing the values of the components up to now is non–decreasing. This ensures that when we have assigned all n components we have found an $\vec{X}_0$ such that $f(\vec{X}_0) \geq E(f(\vec{X}))$.

In [Lub88], Luby observed that using limited independence directly in some parallel algorithms leads to a high processor complexity and introduced a way of combining limited independence with the method of conditional probabilities which led to processor efficient algorithms. His method has been used to derive NC algorithms for fundamental problems such as set discrepancy [BR91,MNN89]. Here we present a new method to combine the use of biased sample spaces and the method of conditional probabilities. We observe that for a large class of problems, direct use of small-bias spaces leads to inefficient NC algorithms, and present a way to combine them with the method of conditional probabilities to get significantly better NC algorithms. Using this we obtain improved NC algorithms for many basic problems such as finding large cuts in graphs, set discrepancy, set cover, $(\Delta+1)$-vertex coloring of graphs and others. In each application our method results in algorithms that are faster than previously known algorithms or match the running time of the best known algorithms with a significant reduction in the processor complexity. We assume the EREW PRAM model throughout, and to describe our results we denote an $O(p(n))$ processor, $O(t(n))$ time NC algorithm as a $(p(n), t(n))$ algorithm.

The first application of our technique is to the classical *set discrepancy* (or *set balancing*) problem.

Definition 5 *(***Set Balancing***): Given a ground set X and subsets $S_1, \ldots, S_n$ of X with $|S_i| \leq \Delta$ for each i, find an assignment $\chi : X \to \{0,1\}$ such that $disc(\chi) \doteq \max_i |\sum_{j \in S_i} \chi(j) - \frac{|S_i|}{2}|$ is* small.

Spencer [Spe87] gave a polynomial–time algorithm to find an assignment χ with $disc(\chi) =$

$O(\sqrt{\Delta \log n})$. NC^3 algorithms to find an assignment χ with $disc(\chi) = O(\Delta^{\frac{1}{2}+\delta} \log n)$, for any fixed $\delta > 0$, were first given in [BR91,MNN89] and small-bias spaces were used to do this in NC^1 [NN90], but with $\Omega(n^{3+\frac{2}{\delta}})$ processors. We derive an NC^1 algorithm with processor complexity $O(n^{1+\frac{2}{\delta}+\delta'})$ for any fixed $\delta' > 0$ which is an improvement of $\Omega(n^{2-\delta'})$ in the processor complexity. This yields improved algorithms for problems like vector–balancing [Spe86], lattice approximation [Rag88], and computing ϵ-nets and ϵ-approximations for range spaces with finite VC-dimension [CM93].

Set balancing is a special case of the following general problem: Let the variables $X_1, X_2, \ldots, X_n$ be unbiased and independent random bits and define $X = \sum_{i=1}^{n^a} f_i(X_{i,1}, X_{i,2}, \ldots, X_{i,b\log}$ where a and b are constants. We wish to find a sample point with $X \geq E[X]$. This general framework was introduced in [BR91] (see also [MNN89]), where an $(n^{a+b}, t\log^3 n)$ algorithm is given (here we assume that each f_i is computable in $O(t)$ time with one processor). In most cases (*e.g.*, set discrepancy), it suffices to find a sample point with $X \geq E[X] - n^{-\Theta(1)}$. In these cases our method yields an $O(n^{a+b+\delta}, t\log n)$ algorithm for any fixed $\delta > 0$, or at the other end of the spectrum, an $(n^{a+b}\log^2 n, t\log^2 n)$ algorithm. Small-bias spaces can be used here to obtain $O(t \log n)$ time NC algorithms, but the processor complexity would be $\Omega(n^{3a+2b}\log^2 n)$. As a corollary, we can derive a faster parallel algorithm for constructing maximal independent sets in hypergraphs of constant dimension, improving the algorithm of [Kel92] without incurring a large processor blowup.

The problem of finding a heavy codeword, which generalizes the classical problem of finding a large cut in a graph, was introduced in [NN90]. Given a matrix $A \in Z_2^{m\times n}$, the problem is to find an $x \in Z_2^n$ with Ax (over Z_2) having at least $\frac{m}{2}$ ones. Small bias sample spaces are used in [NN90] to obtain a $(\min\{m^4n^2, m^3n^3\}$,$\log(m+n))$ algorithm, thus placing the problem in NC. Our method directly yields a range of algorithms for this problem with a continuous processor–time tradeoff: the algorithm with the best processor-complexity is an $(n^2m, \log(n+m)\log m)$ algorithm, and the best time-complexity algorithm is an $(n^2m^{1+\delta}, \log(m+n))$ algorithm, for any fixed $\delta > 0$. The problem of finding a *large cut* in a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, is to find a cut with at least $\frac{m}{2}$ cut edges. NC algorithms of complexity $((n+m), \log^2 n)$ are known for this problem (see, *e.g.*, [Lub88]). Small-bias spaces can be used to obtain an NC^1 algorithm but with a processor complexity of $\Theta(m^3\log^2 n)$. A direct use of pairwise independent sample spaces leads to an NC^1 algorithm with processor complexity mn. In contrast, our method yields an $((m+n)n^\delta, \log n)$ algorithm, for any fixed $\delta > 0$.

We combine our method with some ideas of Alon *et al.* [AN92] to derive improved algorithms for some other basic problems. Given a graph $G = (V, E)$ with $|V| = n$, $|E| = m$ and having a maximum degree Δ, a $(\Delta + 1)$-vertex coloring of the graph can be computed in $((n+m), \log^3 n \log\log n)$ [Lub88]. We improve this to give an $((n+m)n^\delta, \log^2 n)$ algorithm where $\delta > 0$ is any fixed constant. This gives faster algorithms for other problems such as the much harder Δ-vertex coloring problem [PS92]. We use similar ideas to derive a faster algorithm for approximating the maximum acyclic subgraph problem[BS90], incurring a very small processor penalty.

A classical problem in combinatorial optimization is the *set cover* problem: given a ground set X with $|X| = n$, $S_1, \ldots, S_m \subseteq X$, and costs $c : X \to \Re$, the problem is to pick $A \subseteq X$ with minimum cost $c(A)$, subject to $|S_i \cap A| \geq 1$ for each i. The *Balanced set cover* problem requires $|A|$ to be "small" subject to the constraint that $|S_i \cap A| \geq \frac{a|A||S_i|}{n}$

for each i, where a is a constant. Chazelle and Friedman use this abstraction to model certain geometric problems [CF90]. NC approximation algorithms for these problems are provided in [BRS89], whose time complexities we can improve by a factor of $\Theta(\frac{\log^2 n}{\log\log n})$ while incurring a processor penalty of $O(\log^2 n)$.

To summarize, our method yields the fastest known NC algorithms for a large class of problems, without a big processor penalty. This method also yields a potential approach to some open problems, where the conditional estimator is a sum (of a certain type) of $n^{O(polylog(n))}$ terms, each depending on $O(\log n)$ unbiased and independent random bits. A key example of this arises in solving set discrepancy in NC, with discrepancy $O(\sqrt{\Delta \log n})$–a challenging open question [BR91,MNN89]. We show that if a certain small-bias space S is efficiently constructible, then our method can solve the above class of problems in NC. This would be interesting since current methods cannot tackle a superpolynomial number of terms. However, we do not yet know if such a sample space exists.

4.1 Description of the Basic Method

We use the following notation in this section: The set $\{1,\ldots,n\}$ is denoted by $[n]$ and $<a,b>$ denotes the dot product, over Z_2, of the vectors a and b. For a binary vector y, $wt(y)$ denotes its Hamming weight, *i.e.*, the number of components which are 1 in the vector.

Definition 6 *A sample space $S \subseteq \{0,1\}^n$ is said to be ϵ-approximate if when $\vec{X} = (X_1,\ldots,X_n)$ is sampled uniformly from S, for all $I \subseteq [n]$ and for all $b_1,\ldots,b_{|I|} \in \{0,1\}$ we have $|\text{Prob}(\wedge_{i\in I} X_i = b_i) - 2^{-|I|}| \leq \epsilon$.*

Definition 7 *The bias of a binary random variable* X *is* $\text{bias}(X) \doteq \text{Prob}(X = 0) - \text{Prob}(X = 1)$. *Let* α *be a nonzero vector in* Z_2^n *and* $\vec{X} = (X_1, \ldots, X_n)$ *be sampled uniformly from a sample space* $S \subseteq \{0,1\}^n$. *The bias of* S *with respect to* α *is* $\text{bias}(<\vec{X}, \alpha>)$ *and is denoted* $\text{bias}_S(\alpha)$. *If* $|\text{bias}_S(\alpha)|$ *is bounded in magnitude by* ϵ *for all nonzero* α, *then* S *is said to be* ϵ*-biased.*

The biases of a sample space S with respect to the vectors $\alpha \in Z_2^n$ are the Fourier coefficients of the probability distribution D_S induced on $\{0,1\}^n$ by sampling uniformly from S [NN90]. The Fourier bases are the functions f_α, where $f_\alpha(x) = (-1)^{<\alpha, x>}$ for $x \in \{0,1\}^n$. If c_α is the Fourier coefficient with respect to α (*i.e.*, the bias of S with respect to α), then by standard Fourier inversion

$$D_S(x) = 2^{-n} \sum_\alpha c_\alpha f_\alpha(x)$$

for $x \in \{0,1\}^n$. From this it can be shown that if all the Fourier coefficients of D_S are bounded in magnitude by ϵ, then the sample space is ϵ-approximate. NC^1 constructions of sample spaces of size $O(\frac{n}{\epsilon^3})$ with Fourier coefficients bounded in magnitude by ϵ were developed in [NN90]. Subsequently, simpler NC^1 constructions of three different sample spaces of size $\frac{n^2}{\epsilon^2}$ were developed in [AGHP92].

The basic idea behind our construction of approximate sample spaces is the following simple fact (see, *e.g.*, [SV86] for a more general version):

Fact 4.1 *Let* $X_1, \ldots, X_k$ *be independent binary random variables each with bias* ϵ. *Then the random variable* $Y = X_1 \oplus X_2 \oplus \ldots \oplus X_k$ *has bias* ϵ^k.

Proof. We supply an elementary proof for completeness. By definition, Y is 1 iff an odd number of the X_j's have the value 1. Since each X_j has bias ϵ, we have $\text{Prob}(X =$

$1) = \frac{1-\epsilon}{2}$. Thus

$$\text{Prob}[Y = 1] = \sum_{j \text{ odd}} \binom{k}{j} (\frac{1-\epsilon}{2})^j (\frac{1+\epsilon}{2})^{k-j}$$

which is $\frac{1-\epsilon^k}{2}$. Thus this bias of Y is ϵ^k. □

The new construction of small bias sample spaces uses this fact as described in the following lemma.

Lemma 4.1 *Let S' be an ϵ'-biased sample space and let $\vec{X_1}, \ldots, \vec{X_{t(\epsilon)}}$ be sampled independently and uniformly at random from S', where $t(\epsilon) = \frac{\lceil \log(\epsilon)}{\log(\epsilon')\rceil}$. Define $\vec{X} = \vec{X_1} \oplus \cdots \oplus \vec{X_t}$. Then the sample space S of all such $\vec{X}$ is ϵ-approximate.*

Proof. By definition, the bias of $<X, \alpha>$ is the bias of $<\vec{X_1}, \alpha> \oplus \cdots \oplus <\vec{X_t}, \alpha>$. Since the bias of each $<\vec{X_i}, \alpha>$ is bounded in magnitude by ϵ' the lemma follows from Fact 4.1. □

Thus our construction of sample space S is to start with a sample space S' of much larger bias and then to XOR independent samples from S'. The space constructed as above will, in general, be larger than those obtained by direct constructions of ϵ-approximate spaces. However when we actually use this method in designing algorithms we will make the different samples from the space S' using the method of conditional probabilities. Since the base sample space S' has $|S'| \ll |S|$, this will yield more efficient algorithms since we have fewer points to try out while derandomizing.

Since the construction uses a known construction of small biased sample spaces to bootstrap itself, we describe a known construction of small bias sample space for completeness. In our algorithms, unless otherwise stated we will use the following construction given in [AGHP92] to bootstrap itself.

Theorem 4.1 [AGHP92] *Let* bin*:* $GF(2^m) \mapsto \{0,1\}^m$ *be the standard binary representation of the field of* 2^m *elements which satisfies* bin*(0)=* 0^m *and* bin*(*$u+v$*)=*bin*(u)*$\oplus$ bin*(v). Given two field elements* x *and* y *define the sequence* $r(x,y) = r_0(x,y)\ldots r_{n-1}(x,y)$ *where* $r_i(x,y) = <\text{bin}(x^i), \text{bin} y>$. *The sample space consisting of the string* $r(x,y)$ *for all choices of* x *and* y *is* $\frac{n-1}{2^m}$ *biased. Thus there exists an* ϵ*–biased sample space of size* $O(\frac{n^2}{\epsilon^2})$.

Similar definitions can be made for sample spaces that are k-wise ϵ-approximate. In definition 6 if the sample space satisfies the condition for all sets $I \subseteq [n]$ of size at most k, then the sample space is said to be (k,ϵ)-approximate. A sample space S is said to be *k-wise ϵ-biased* if $|bias_S(\alpha)| \le \epsilon$ for all nonzero $\alpha \in Z_2^n$ with $wt(\alpha) \le k$. A k-wise ϵ-biased space is (k,ϵ)-approximate[NN90]. Sample spaces of size $O(\min\{\frac{k\log n}{\epsilon^3}, \frac{(k\log n)^2}{\epsilon^2}\})$ which are k-wise ϵ-biased can be constructed in NC^1 [NN90,AGHP92,ABN$^+$92]. If S' is k-wise ϵ'-biased in Lemma 4.1, then the resulting space S is k-wise ϵ-biased. Thus our construction extends directly to yield k–wise ϵ–biased sample spaces.

4.2 Direct Applications of the Method

Throughout this section we use $G=(V,E)$ to denote a graph on n vertices and m edges, with maximum degree Δ. $\delta, \delta', \delta'', \delta_1, \delta_2$ etc. denote arbitrarily small positive constants.

4.2.1 Cuts in Graphs and the heavy codeword problem

As a straightforward application of our method, we consider the *heavy codeword* problem [NN90]: given $A \in Z_2^{m\times n}$, find an $x \in Z_2^n$ such that Ax (over Z_2) has at least $\frac{m}{2}$ ones. In the following a_i denotes the ith row of A, and we assume that the a_i's are nonzero.

As pointed out in [NN90], if x is picked from an ϵ-biased space S with $\epsilon = \frac{1}{2m}$, then

$$\mathrm{E}[wt(Ax)] = \sum_{i=1}^{m} Pr(<a_i, x> = 1) =$$

$$\sum_{i=1}^{m} \left(\frac{1 - \mathrm{bias}(<a_i, x>)}{2} \right) \geq m\,(1/2 - \epsilon/2)$$

and thus, there exists an $x \in S$ with $wt(Ax) \geq \lceil m\,(1/2 - \epsilon/2) \rceil = \lceil m/2 \rceil$. But, an $\frac{1}{2m}$-biased space has $\min\{m^3 n, m^2 n^2\}$ points and for each point the obvious algorithm needs mn processors to compute Ax. Thus, direct use of small-bias spaces results in high processor complexity. Instead, we start with an m^{δ}-biased sample space S_0 and pick x, as in Section 4.1, to be the bit-wise XOR of $\ell = \lceil \frac{\log 2m}{\log m^{\delta}} \rceil = O(1)$ many *independent* samples $Y_1, Y_2, \ldots, Y_\ell$ from S_0.

We show that the method of conditional probabilities can be used to find a "good" sequence $Y_1^*, \ldots, Y_\ell^*$ *efficiently* in NC. Assume that we have already fixed the values of $Y_1^*, \ldots, Y_{j-1}^*$. Define the conditional estimator at stage j to be

$$f_{j-1}(Y_1^*, \ldots, Y_{j-1}^*) \doteq \sum_{i=1}^{m} \frac{(1 - \mathrm{bias}(<a_i, x>|Y_k = Y_k^*, 1 \leq k \leq j-1))}{2}$$

We want a $Y_j^* \in S_0$ which maximizes $f_j(Y_1^*, \ldots, Y_j^*)$. We need only show that $f_j(Y_1^*, \ldots, Y_j^*)$ can be computed efficiently for all points $Y_j^* \in S_0$. Let $b = \bigoplus_{k=1}^{j} Y_k^*$. Thus, $x = b \oplus z$, where $z = Y_{j+1} \oplus \cdots \oplus Y_\ell$ with $Y_{j+1}, \ldots, Y_\ell$ picked uniformly and independently from S_0. Thus, $\mathrm{bias}(<a_i, x>|Y_k = Y_k^*, 1 \leq k \leq j)$ equals $\mathrm{bias}(<a_i, z>)$ if $<a_i, b> = 0$, and $-\mathrm{bias}(<a_i, z>)$ otherwise. Note the crucial role of the independence of the different samples. From Lemma 4.1, $\mathrm{bias}(<a_i, z>) = (\mathrm{bias}_{S_0}(a_i))^{\ell - j}$. The algorithm is described in detail in Figure 4.1.

The size of S_0 is $O(m^{3\delta} n)$ and hence we can precompute $\mathrm{bias}_{S_0}(a_i)$ for all the a_i's using $mn|S_0|$ processors in $O(\log n + \log m)$ time. At each of the ℓ stages we have

Construct a sample space S_0 which is $\epsilon' = m^\delta$ biased.

For each i in parallel precompute $\text{bias}_{S_0}(a_i)$

For j from 1 through $\ell = \lceil \frac{\log 2m}{\log m^\delta} \rceil$ do

  For each point y_j in S_0 in parallel do

    Compute $b = (\bigoplus_{k=1}^{j-1} Y_k^*) \oplus y_j$.

    For each i in parallel do

      If $< a_i, b > = 0$then let $\text{bias}_i = (\text{bias}_{S_0})^{\ell - j}$

        else let $\text{bias}_i = -(\text{bias}_{S_0})^{\ell - j}$

  Let Y_j^* to be the value of y_j which maximizes $\sum_1^n \frac{1-\text{bias}_i}{2}$

Output $\bigoplus_{k=1}^{\ell} Y_k^*$.

Figure 4.1: Efficient Algorithm for the heavy codeword problem

$|S_0|$ choices for Y_j^* and $f_j(Y_1^*, \ldots, Y_j^*)$ can be evaluated in $O(\log(m+n))$ time using $O(mn|S_0|)$ processors. Alternatively, we could choose S_0 of constant bias and the same idea gives an $(n^2 m, \log(m+n)\log m)$ algorithm.

Theorem 4.2 *There exist* $(n^2 m^{1+\delta}, \log(m+n))$ *and* $(n^2 m, \log(m+n)\log m)$ *algorithms for the* heavy codeword *problem.*

We reiterate the points which led to efficiency: (a) the search space $|S_0|$ has $|S_0| \ll \min\{m^4 n^2, m^3 n^3\}$, and (b) we directly compute a good point x, avoiding the costly matrix–vector multiplications.

Given a graph G, similar ideas work to find a cut with at least $\frac{m}{2}$ edges. If each vertex picks a bit to decide which side of the cut it lies on, and if the bits come from a 2-wise $\frac{1}{2m}$–biased sample space, then $\lceil E[\#\text{ cut edges}]\rceil \geq m/2$. Direct use of biased spaces yields an $(m^3 \log^2 n, \log n)$ algorithm. A better algorithm can be obtained by using a sample space which guarantees pairwise independence and this results in a $(mn, \log n)$ algorithm. In contrast using our method we obtain

Theorem 4.3 *There exists an algorithm for the large cut problem which runs in time* $((n+m)\min\{\log n/\epsilon^3, \log^2 n/\epsilon^2\}$ *using* $\frac{\log m}{\log(\frac{1}{\epsilon})}\log n$ *processors for any* $\epsilon = \epsilon(n,m) < 1$. *In particular, it can be solved by an* $((n+m)n^\delta, \log n)$ *algorithm.*

4.2.2 Applications using biases to compute probabilities

We now outline a more involved application where we use the biases of the base distribution to compute probabilities of events by Fourier inversion. Given subsets $S_1, \ldots, S_n$ of a set X with $|S_i| \leq \Delta$ for each i, the *set discrepancy* problem is to find a $\chi : X \mapsto \{0,1\}$ such that $disc(\chi) \doteq \max_i |\chi(S_i) - \frac{|S_i|}{2}|$ is "small", where $\chi(S_i) = \sum_{j \in S_i} \chi(j)$. It is shown in [BR91,MNN89] that if we choose $k = 2\lceil\frac{\log(2n)}{\delta\log\Delta}\rceil$ and choose the assignments $\chi(j), j \in X$ $k - wise$ independently then

$$\text{Prob}[\ \ (\exists i)\ |\chi(S_i) - \frac{|S_i|}{2}| \geq \Delta^{0.5+\delta}\sqrt{\log n}\ \ ]$$

$$\leq \sum_{i=1}^{n} \frac{E[(\chi(S_i) - \frac{|S_i|}{2})^k]}{(\delta^{0.5+\delta}\sqrt{\log n})^k} < 1. \tag{4.1}$$

This is used to derive NC^3 algorithms which find a χ with $disc(\chi) \leq \Delta^{0.5+\delta}\sqrt{\log n}$. It is shown in [NN90] that (4.1) holds even if the assignments $\chi(j), j \in S$ are k–wise ϵ-biased for $\epsilon < \frac{1}{2n^{1+\frac{1}{\delta}}}$, leading to an $(\Omega(n^{3+\frac{2}{\delta}}), \log n)$ algorithm. We significantly improve the processor complexity using Fourier coefficients to compute probabilities. Expanding $E[(\chi(S_i) - \frac{|S_i|}{2})^k]$ gives $O(n^{\frac{2}{\delta}})$ terms, each of the form $c \cdot Pr(X_{i_1} = \cdots = X_{i_t} = 1)$, with $t \leq k$ and c a constant. We know how to compute probabilities of events once we we know the Fourier coefficients as shown in Section 4.1. In this case we can express the term $c \cdot Pr(X_{i_1} = \cdots = X_{i_t} = 1)$ as the sum of $2^t = n^{o(1)}$ *bias* terms, using the Fourier bases. Now exactly as in the case of the algorithm for the heavy codeword problem we can start with a k-wise $n^{-\delta''}$-biased S_0 and pick a "good" χ using $O(1)$ samples from S_0. At each stage we choose a point in S_0 which minimizes the expectation which we have expressed in terms of the biases. The number of sample points we need to try at each stage $|S_0| = n^{O(\delta'')}$ and the number of bias terms we need to consider is $n^{\frac{2}{\delta}+o(1)}$. The algorithm runs in time $O(\log n)$ and the number of processors we need for each stage is $O(n^{1+\frac{2}{\delta}+\delta'})$. Thus we can obtain the following.

Theorem 4.4 *There exists an $(n^{1+\frac{2}{\delta}+\delta'}, \log n)$ algorithm to find a χ for the set discrepancy problem with $disc(\chi) \leq \Delta^{0.5+\delta}\sqrt{\log n}$, for any fixed $\delta, \delta' > 0$.*

The set discrepancy abstraction has been used to solve other problems such as vector balancing [Spe86], lattice approximation[Rag88](see also the description in [MNN89]) and for computing ϵ-nets and ϵ-approximations for range spaces with finite VC–dimension [CM93]. Theorem 4.4 implies improved NC algorithms for all these problems. Our method also yields improvements in processor complexity for algorithms which find an assignment χ with $disc(\chi) = O(\sqrt{\Delta \log n})$. We can derive a $(\Delta^{\Theta(\log n)}, \log n)$ algorithm with the constant in the exponent of the processor complexity roughly half of what a direct implementation using regular ϵ-biased spaces can yield.

Set discrepancy has been captured as part of a more general framework in [BR91], and we can extend the above method to the general framework to derive improved algorithms: Suppose $X_1, \ldots, X_n$ are unbiased and independent random bits and let

$$X = \sum_{i=1}^{n^a} f_i(X_{i,1}, X_{i,2}, \ldots, X_{i,b\log n}).$$

We wish to find a sample point with $X \geq E[X]$, where a and b are constants.

An $(n^{a+b}, t\log^3 n)$ algorithm is presented for this problem in [BR91] where each f_i computable in $O(t)$ time with one processor. In many cases, such as the set discrepancy problem above, if the X_is are $b\log n$-wise,ϵ-biased with $\epsilon = n^{-O(1)}$, we can find a sample point with $X \geq E[X] - n^{-\Theta(1)}$ which suffices. We can first express the sum in terms of n^{a+b} bias terms using elementary harmonic analysis[BR91]. Then, if we start with a sample space S_0 with $n^{-\delta}$ bias and use the same approach as for set discrepancy we can get an $(n^{a+b+\delta}, t\log n)$ algorithm. On the other extreme, if we start with a sample space which is of constant bias we can get an $(n^{a+b}\log^2 n, t\log^2 n)$ algorithm. In contrast, directly plugging in the required small–bias sample space only gives an $(\Omega(n^{3a+2b}), t\log n)$

algorithm. As a corollary we get a faster parallel algorithm to find maximal independent sets in hypergraphs of constant dimension, improving that of [Kel92] incurring too much of a processor penalty.

As another example we can get from our improved algorithm for the general abstraction, an improved algorithm for a special case of the set discrepancy problem. If $\Delta = \log^{O(1)} n$ in the set discrepancy problem, then an $(n^2 \log^{O(1)} n, \Delta^2 \log^3 n)$ algorithm is given in [BR91] to find χ with $disc(\chi) = O(\sqrt{\Delta \log n})$. The original algorithm uses the above general framework as a subroutine and replacing the original algorithm with ours results in the following improvement. We refer the reader to [Ber90,BR91] for details.

Theorem 4.5 *If $\Delta = \log^{O(1)} n$ in the set discrepancy problem then there exist an $(n^{2+\delta}, \Delta^2 \log n)$ algorithm to find χ with $disc(\chi) = O(\sqrt{\Delta \log n})$, for any fixed $\delta > 0$.*

4.3 Handling multivalued variables

In this section we describe a number of applications where the random variables of interest are uniform over $\{0,1\}^q$ for some $q > 1$. By combining our basic method with some ideas of [AN92] we get improved parallel algorithms for various problems. Some of these problems can also be solved using the general framework described at the end of the previous section. However the solutions we give here are better than those obtained as instantiations of the general framework.

4.3.1 The PROFIT/COST problem

The first application we describe is the "General PROFIT/COST Problem" defined in [Lub88]: Given a graph $G = (V, E)$ with functions PROFIT, COST: $\{0,1\}^q \rightarrow \Re$ and

$\ell : V \rightarrow \{0,1\}^q$, define

$$Benefit(\ell) \doteq \sum_{v \in V} \mathrm{PROFIT}(\ell(v)) - \sum_{u,v \in E} \mathrm{COST}(\ell(u), \ell(v)).$$

The problem is to find an $\hat{\ell}$ such that $Benefit(\hat{\ell}) \geq E[Benefit(\ell)]$. Note that we can assume that the labels $\ell(v)$ are uniform and pairwise independent over $\{0,1\}^q$ to compute the expected benefit. This abstraction models many problems such as $(\Delta + 1)$-vertex coloring in graphs [Lub88]. To derive an $(n+m)$-processor algorithm, where $n = |V|$ and $m = |E|$, Luby assumes that *pessimistic estimator functions* $Profit$ and $Cost$ exist which can be computed in time t with one processor, and replaces PROFIT and COST by these functions. We refer the reader to [Lub88] for details on the definition of these estimator functions. The strategy used in [Lub88] is to set one bit of each of the labels $\hat{\ell}(v)$ per stage without using too many processors. This results in an algorithm using $O(n+m)$ processors and which runs in $O(q(t + \log n \log\log n) + \log^2 n \log\log n)$ time.

Our goal is to derive an $O(t + \log n)$ time algorithm to find $\hat{\ell}$ with $Benefit(\hat{\ell}) \geq E[Benefit(\ell)] - n^{-\Theta(1)}$ for those instantiations of the the general problem where the estimator functions $Profit$ and $Cost$ are bounded in value by a polynomial. We need this extra requirement since we will use labels which are pairwise ϵ biased and we can handle only inverse polynomial values for ϵ. A direct approach following the earlier algorithms would be to start with a $2q$-wise $n^{-\Theta(1)}$-biased nq-length bit string and assign labels by partitioning the nq bits into n labels. However the methods of the previous section require us to compute probabilities by first converting them to the appropriate bias terms and the large number of Fourier coefficients increases the processor complexity by a 2^{2q} factor. Instead we adopt the following strategy: we set $\delta_1 q$ bits of each of the labels $\hat{\ell}(v)$ all at once in each stage, and repeat δ_1^{-1} times. In each stage, sampling from a $2\delta_1 q$-wise $n^{-\Theta(1)}$-biased $n\delta_1 q$-length sample space S_1 suffices. The sample space S_1 is constructed by starting from an $n^{-\delta_2}$-biased sample space S_0 and taking $O(1)$ samples, as in section 4.2.1. In each stage we have to find a point in S_1 which maximizes a sum of $(n+m)2^{2\delta_1 q}$ bias terms which we do by the method of conditional probabilities by sampling $O(1)$ times from the space S_0. Thus each stage can be implemented using $(n+m)2^{O(\delta_1 q)}|S_0|$ processors and the running time is $O(t + \log n + q)$ for each stage. Plugging in the size of S_0 from the standard construction of [AGHP92] we get the following improvement.

Theorem 4.6 *For any fixed $\delta > 0$, there exists an $((n+m)2^{O(\delta q)}n^{O(\delta)}, t + \log n + q)$ algorithm to find a labeling $\hat{\ell}$ with $Benefit(\hat{\ell}) \geq E[Benefit(\ell)] - n^{-\Theta(1)}$ in the General PROFIT/COST problem for the case that the functions $Profit$ and $Cost$ are bounded in value by a polynomial.*

As instantiations of this general framework we can derive the following as corollaries which are obtained from the above theorem and results in [Lub88] and [PS92].

Corollary 4.1 *$(\Delta + 1)$-vertex coloring is solvable in $((n+m)n^{\delta}, \log^2 n)$, for any fixed $\delta > 0$.*

Corollary 4.2 *For any fixed $\delta > 0$, there exists an $((n+m)n^{\delta}, \log^4 n/\log\Delta)$ algorithm for Δ-vertex coloring.*

4.3.2 The large d–partite subhypergraph problem

A hypergraph $H = (V, E)$ consists of a vertex set V and a set of edges which are arbitrary subsets of V. Given a hypergraph with each edge having exactly d vertices the large d-partite subhypergraph problem considered by [EK68,ABI86] is to color V with d colors such that the number of edges which have the d different colors is at least $|E|d!/d^d$. Alon *et al.*[ABI86] give an NC algorithm for this problem for the case when d is a constant. Let $D = \min\{d, \log|E|\}$. Berger and Rompel[BR91] derive a deterministic $(|E|2^D, D\log n(\log n + \log|E|)\log|E|\log\log|E|)$ algorithm for this problem. They first show by analysis that if each vertex is labeled with a randomly chosen $\ell = O(\log(|E|)\log\log(|E|))$ bit label to define its color there is some coloring which produces a large d-partite subhypergraph. Their deterministic algorithm then uses the general framework described in Section 4.2.2 as a subroutine to set the label bit by bit making sure that each stage the expectation of the number of d colored edges is maximized. We can replace their invocation of the general framework by our method and choosing each bit by sampling from a space of the appropriate bias we can get the following

Theorem 4.7 *For any fixed constant $\delta > 0$ there exists an algorithm to solve the large d-partite subhypergraph problem which uses $|E|^{1+\delta}2^{O(D)}$ processors and running in time $D\log^2|E|(\log\log|E|)^2$.*

4.3.3 The maximum acyclic subgraph problem

For the above examples we just directly plugged in our improvement to the generalization of the set discrepancy into the previously known algorithm of [Ber90] to get an improved algorithm. Consider the Maximum Acyclic Subgraph problem: Given a directed graph $G = (V, A)$ we wish to find a subset $\hat{A}$ of the arcs such that the subgraph $G' = (V, \hat{A})$ is acyclic. The problem of finding the optimal sized $\hat{A}$ is known to be NP–complete.

Berger[Ber90] derives an NC algorithm which finds a subset $\hat{A}$ of size at least

$$\frac{|A|}{2} + \Omega(\sum_{i=1}^{n} \sqrt{deg(i)} + \sum_{i=1}^{n} |d_{out}(i) - d_{in}(i)|)$$

where $deg(v) = d_{out}(v) + d_{in}(v)$. The essential idea in their algorithm is to assign a randomly chosen label in $[1, \sigma]$ to each of the vertices. A vertex is processed as follows: Count the outdegree of the vertex only considering the edges to vertices with larger labels and similarly count the indegree from vertices with larger labels. Choose for $\hat{A}$ all inedges (outedges) from (to) larger numbered neighbors if the indegree (outdegree) is higher breaking ties arbitrarily. The resulting set of arcs clearly defines an acyclic graph and using the fourth moment method it is shown that if the labels are assigned even 5–wise independently, with $\sigma = n$, the expectation is at least the above sum.

To derandomize this algorithm they assign the labels of size $\log \sigma$ to each vertex bit by bit all at once and at each stage they use the general framework to find an assignment which maximizes the expectation of the number of the edges in the acyclic graph. This results in a $(n\Delta^4, \log^2 n \log \sigma)$ algorithm to approximate the maximum acyclic subgraph problem. We could replace their invocation of the general framework by ours and this would result in an algorithm of complexity $(n^{1+\delta}\Delta^4, \log n \log \sigma)$ for any fixed $\delta > 0$. However we use a strategy similar to the one for the general profit/cost problem: In each stage we set $\delta_1 \log \sigma$ bits of the labels of each of the vertices all at once. In each stage we need to sample from a sample space which is $(5\delta_1 \log \sigma)$–wise $n^{-O(1)}$ biased sample space and we wish to optimize the sum of $n\Delta^4$ functions each depending on at most $5\delta_1 \log \sigma$ bits. First we will reduce this to maximizing $n\Delta^4 \times 2^{5\delta_1 \log \sigma}$ bias terms. In each stage we start with a sample space which is $(5\delta_1 \log \sigma)$–wise $n^{-\delta_2}$ biased and take the XOR of $O(1)$ samples. Each stage uses $n\Delta^4 \times 2^{5\delta_1 \log \sigma} n^{O(\delta_2)}$ processors and runs in time $O(\log n + \log Delta + \log \sigma)$ time. There are a constant number of such stages and plugging in $\sigma = n$ we get the following improved algorithm. Since the details are straightforward from the analysis in [Ber90] and the algorithm is very similar to our algorithm for the general Profit/Cost problem we will not describe the details of the algorithm.

Theorem 4.8 *There exists a $(n^{1+\delta}\Delta^4, logn)$ algorithm to find an acyclic subgraph in a given graph with at least*

$$\frac{|A|}{2} + \Omega(\sum_{i=1}^{n} \sqrt{deg(i)} + \sum_{i=1}^{n} |d_{out}(i) - d_{in}(i)|)$$

edges.

4.3.4 Set cover and Balanced set cover

See Section 4 for the definitions of the *set cover* and *balanced set cover* problems for a given hypergraph $H = (V, E)$. For the balanced set cover problem, we combine some ideas from the previous section and use the following two observations to get improved algorithms:

1. The benefit function which is maximized in the algorithm of [BRS89] has a term of the form $\sum_{e \in E} w_e (\sum_{i \in e} X_i - \sum_{i,j \in E, i \neq j} X_i X_j)$ where w_e is a defined weight for the edge e and the variables X_i are binary random variables. This can be rewritten as $\sum_{e \in E} w_e (2 \sum_{i \in e} X_i - (\sum_{i \in e} X_i)^2)$, which can be computed fast using few processors.

2. The variables X_i in the above expression are indicator variables for a $\log n$ sized binary label being all zeros. As before, we start by fixing one bit of all the labels

all at once. However, many terms in the conditional benefit vanish as we proceed and so, we can fix larger and larger sequences of the random bits as we progress.

We refer the reader to the algorithm of [BRS89] for details of the algorithm. Similar observations also hold for their set cover algorithm. Using these and the method of the previous section we obtain the following.

Theorem 4.9 *For any fixed $\delta < 1$, there exists an algorithm using $(\sum_{e \in E} |e| + n) \log^2 n$ processors and running in $\frac{\log^2 n \log m \log^2(nm) \log\log n}{\delta^6}$ time, to find a set cover of weight at most $(1+\delta)(1+\ln\Delta)$ times the weight of an optimal cover in the given hypergraph.*

Theorem 4.10 *For balanced set cover there is an algorithm which uses $(\sum_{e \in E} |e| + n) \log^2 n$ processors and which runs in $\log^2(nm) \log\log n$ time to produce a cover R of size $O(\frac{\log(nm)}{\alpha)}$ where $\alpha = min_{e \in E} \frac{|e|}{n}$, such that $\forall e \in E$ $|R \cap e| \geq \frac{|e||R|}{16n}$.*

4.4 Potential approach to some open problems

Consider the following abstract problem: Let $X_1, \ldots, X_n$ be independent random variables and define an *abstract estimator* to be a function of the form

$$X = \sum_{i=1}^{\log^c n} a_i \sum_{A \subseteq [n], |A| = i} \text{bias}(\oplus_{j \in A} X_j).$$

We wish to find a sample point with $X \leq E[X]$. An obvious instantiation of this framework is the set discrepancy problem where we wish to find an assignment χ with $disc(\chi) = O(\sqrt{\Delta \log n})$ using the $O(\log n)$-th moment method *i.e.* using variables which are $O(\log n)$ wise independent[BR91,MNN89]: For this application we have a sum of n functions of this type, with $c = 1$. All the currently known methods fail to solve this problem in NC since they need one processor per term and there are $n^{\Theta(\log^c n)}$ terms. In most cases, it suffices to find a sample point with $X \leq E[X] + n^{-\Theta(1)}$. Therefore we can do with a source of random bits which is a $\log^c n$-wise $n^{-\Theta(polylog(n))}$-biased space S would suffice. However, it can be shown that the size of such a sample space must be superpolynomial [AGHP92].

Suppose there exists a $\log^c n$-wise ϵ-biased space S_0 such that

(a) $\epsilon =$ constant and hence $|S_0| = n^{O(1)}$ and

(b) For any $\ell \leq \log^c n$ and any $S_1, S_2 \subseteq [n]$ with $|S_1| = |S_2| = \ell$, $\text{bias}(\oplus_{i \in S_1} X_i) = \text{bias}(\oplus_{i \in S_2} X_i)$ $(\leq \epsilon)$, for a random $\vec{X} \in S_0$.

If such a sample space exists then we claim that this problem is in NC. As before we take $\log^{O(1)} n$ samples $Y_1, Y_2, \ldots$ from S_0 using our method. Suppose we have picked $Y_1, \ldots, Y_i$ with $\oplus_{j=1}^{i} Y_j = b$. Then, the conditional benefit is easily computed since, for each $\ell \leq \log^c n$, we just need to know the *number* of sets $S \subseteq [n]$ with $|S| = \ell$ and $\oplus_{j \in S} b_j = 1$, which can be combinatorially evaluated.

Theorem 4.11 *The* abstract estimator *problem is in NC if there is an efficiently constructible S_0 satisfying (a) and (b). The special case $c = 1$ solves set discrepancy in NC.*

However, we do not even know if such a sample space S_0 exists! A $\log^c n$-wise independent space satisfies condition (b), but is too big. However property (b) seems too stringent: we leave as open the questions whether exact equality is necessary in condition (b).

4.5 Discussion

In this chapter we devised a new probabilistic technique: the construction of a sample space whose Fourier coefficients are close to those of a independent random source on n random variables. Though the size of the sample space so constructed is larger than those of previous constructions its definition is particularly amenable to the application of the method of conditional probabilities. Using this we derived efficient algorithms for several combinatorial optimization problems. For a large class of problems our method yields algorithms which are faster than previously known algorithms or match the running time of the best known algorithms with a significant reduction in the processor complexity.

Chapter 5

Construction of small sample spaces satisfying given constraints[1]

In the previous chapter we described a new construction of a sample space which satisfied the constraint of being close to an independent distribution on n binary random variables. In this chapter we consider the abstract problem of constructing sample spaces to satisfy given constraints. These constructions are motivated by their potential applications to transforming probabilistic algorithms into deterministic ones: Given a probabilistic algorithm, by analysis we can usually identify a set of constraints on the random bits input to the algorithm such that the algorithm performs equally well under any source satisfying these constraints. We can then construct a sample space which satisfies the required constraints and a deterministic algorithm is obtained by running the probabilistic algorithm on each point in the sample space. The efficiency of the deterministic

[1] The results in this chapter were first presented in [CRS94] and [CC94]. Section 1 describes joint work with Drs. Aravind Srinivasan and Pankaj Rohatgi. The results in section 2 were obtained jointly with Prof. Jin–yi Cai of the State University of New York, Buffalo.

algorithm is determined by the size of the sample space. Thus a crucial concern to us in this chapter is to find small sample spaces satisfying given constraints and how close they are to the lower bounds on the size of such sample spaces.

The first problem we address is the generalization of the problem in the previous chapter. We construct sample spaces which approximate the distribution of n independent *multivalued* random variables. With the same motivation of applying such constructions to deriving efficient deterministic algorithms from their randomized counterparts this problem was first considered by Even *et al.* [EGL$^+$92]. They give three different constructions of such sample spaces each of which is suitable for different ranges of the input parameters. In this chapter we shall give a construction such that the resulting sample space is much smaller than all the constructions of [EGL$^+$92]. Though this space is smaller than all the previously known constructions this is still far from the known lower bounds for the size of such sample spaces.

As we have seen in the previous chapter, a very important tool in converting probabilistic algorithms to deterministic ones is the idea of limited independence of random variables. It can be shown that sample spaces defining a k–wise independent distribution must contain at least $\binom{n}{1} + \ldots + \binom{n}{\lfloor k/2 \rfloor}$ points[ABI86,ASE92,CGH$^+$85]. Thus, polynomial size constructions are possible only for constant k. Schulman[Sch92] observes that for several algorithms we do not need independence in all the $\binom{n}{k}$ subsets of indices and constructs sample spaces which are independent only for specified index sets of k elements. He gives constructions of sample spaces which are independent only on specified index sets and uses these constructions to get efficient algorithms for several problems. We will investigate this problem of constructing sample spaces independent on only specified sets of index sets in detail. In Section 2 we prove improved lower and upper bounds on the size of sample spaces which are required to be independent on specified neighborhoods. Our new constructions yield sample spaces whose size is smaller than previous constructions due to Schulman[Sch92]. Our lower bounds generalize the known lower bounds of [ABI86,CGH+85,ASE92]. In obtaining these bounds we examine the possibilities and limitations of amplifying limited independence by fixed functions. We show that in general independence can not be amplified from k–wise independence to $(k+1)$–wise independence. Finally, we enumerate all possible logical consequences of pair–wise independent random bits.

5.1 Approximating general distributions

Small spaces which approximate the distribution of independent and uniform binary random variables have become a key tool in derandomization. With this motivation, Even *et al.* [EGL+92] consider the general problem of approximating the distribution of independent multivalued random variables.

Definition 8 *Let* $X_1, \ldots, X_n \in \{0, \ldots, m-1\}$ *be* independent *random variables with* arbitrary *individual distributions and joint distribution* $\mathcal{D}$*. A sample space* S *is a* (k, ϵ)*-*approximation *of* $\mathcal{D}$ *if, for* $\vec{Y} = (Y_1, \ldots, Y_n)$ *sampled uniformly from* S*, and any set* I *of at most* k *indices we have for all* $a_1, \ldots, a_{|I|}$ *in* $[0, m-1]$

$$|Pr_{\vec{Y} \in S}(\wedge_{i \in I} Y_i = a_i) - \Pi_{i \in I} Pr_{\mathcal{D}}(X_i = a_i)| \leq \epsilon.$$

The case $m = 2$ with $\mathcal{D}$ being the uniform distribution is handled by [NN90,AGHP92, ABN+92]. We are interested in a *single* sample space which approximate *all* such $\mathcal{D}$.

Three such constructions were given in [EGL$^+$92], each suitable for different ranges of the parameters. We now present two constructions whose size either matches or is smaller than those of [EGL$^+$92].

To construct (k, ϵ)-approximate sample spaces, we can discretize the problem and only consider points in the n-dimensional mesh $[0, \frac{4k}{\epsilon} - 1]^n$. It is shown in [EGL$^+$92] that this approximation does not result in a big error. So we will assume that the distribution $\mathcal{D}$ which we wish to approximate has probabilities which are rationals with denominator $\frac{4k}{\epsilon}$. Now an event of the form $Y_j = a_j$ corresponds to an interval in dimension j and the event $\wedge_{j=1}^{k} Y_{i_j} = a_{i_j}$ corresponds to a rectangle in k dimensions. Let $\Re_n^k$ denote the set of all k dimensional rectangles. For $R \in \Re_n^k$ let $\Delta_S(R) = |\mathrm{vol}(R) - \frac{|S \cap R|}{|S|}|$. A sample space S is (k, ϵ)-approximate if for all $R \in \Re_n^k$ we have $\Delta_S(R) \leq \epsilon$ which we denote as $\Delta_S(\Re_n^k) \leq \epsilon$ [EGL$^+$92]. The main result of our constructions is the following.

Theorem 5.1 *There is an explicitly constructible sample space* $S \subseteq [0, \frac{4k}{\epsilon} - 1]^n$*, of size polynomial in* $\log n$*,* $\frac{1}{\epsilon}$*, and* $\min\{2^k, k^{\log(\frac{1}{\epsilon})}\}$*, such that* $\Delta_S(\Re_n^k) \leq \epsilon$*.*

In their original paper Even *et al* [EGL$^+$92] describe an efficient method to explicitly construct three sample spaces $S_1, S_2, S_3 \subseteq [0, \frac{4k}{\epsilon} - 1]^n$ to achieve the above with $|S_1| = \mathrm{poly}(\log n, 2^k, \frac{1}{\epsilon})$ and $|S_2| = \mathrm{poly}(\log n, k^{\log(\frac{1}{\epsilon})}, (\frac{1}{\epsilon})^{\log(\frac{1}{\epsilon})})$. The sample space S_3 of size $\mathrm{poly}(n^{\log n}, n^{\log(\frac{1}{\epsilon})})$ guarantees $\Delta_{S_3}(\Re_n^n) \leq \epsilon$. Thus, our construction either matches or is better in size than all three of [EGL$^+$92]. For instance if $k = \log^b n$ and $\epsilon = n^{-\Theta(1)}$ with $b \geq 2$, our construction has size $n^{O(\log\log n)}$, while those of [EGL$^+$92] have size $n^{O(\log n)}$.

5.1.1 The first construction

Consider a rectangle $R \subseteq [0, \frac{4k}{\epsilon} - 1]^n$ which is nontrivial (*i.e.*, not equaling $[0, \frac{4k}{\epsilon} - 1]$) in at most k dimensions, say dimensions $1, \ldots, k$. To define the random variables $Y_1, \ldots, Y_n$ which define the small space, we sample uniformly from a space which (k', ϵ')-approximates the distribution of $n \log(\frac{4k}{\epsilon})$ independent and uniform binary random variables for suitable k' and ϵ'. Then Y_i is assigned the value of the i^{th} block of $\log(\frac{4k}{\epsilon})$ bits. We sketch a proof that choosing k' and ϵ' suitably gives $\Delta = \Delta_S(R) \leq \epsilon$ with $|S| = \text{poly}(\log n, 1/\epsilon, k^{\log(\frac{1}{\epsilon})})$, for a range of values for k.

To analyze Δ, we partition R into disjoint rectangles of a special form. For any integer m, define the m-*canonical* intervals to be intervals of the form $[k2^\alpha, k2^\alpha + 2^\alpha - 1]$ where $k, \alpha \geq 0$ and $k2^\alpha + 2^\alpha - 1 \leq m - 1$. Any interval $[a, b]$ in the range $[0, m-1]$ can be decomposed into at most $c \log m$ canonical intervals, for some constant c (see [BW80, PS85]). A rectangle is canonical if its projection along each dimension is a canonical interval. Any rectangle in t dimensions can be decomposed into $(c \log m)^t$ canonical rectangles. The crucial property we use is that a number lies in a canonical interval iff the binary expansion of the number has a specific sequence of most significant bits. Directly decomposing R results in too many canonical rectangles and to reduce the dimension somewhat we use Lemma 5.1, which can be observed from the proof of Theorem 2 of [EGL+92] (see also Theorem 2.6 in [SSS93]).

Lemma 5.1 *For* $\{0,1\}$ *random variables* $X_1 \ldots X_n$ *with any joint distribution* D *and* $Pr(X_i = 1) = p_i$,

$$|Pr_D(\bigwedge_{i=1 \ldots n} X_i = 1) - \prod_{i=1,\ldots,n} p_i| \leq \epsilon^2 +$$

$$\sum_{\ell=1}^{2\log(\frac{1}{\epsilon})} \sum_{i_1<\cdots<i_\ell} |Pr_D(\bigwedge_{j=1,\ldots,\ell} X_{i_j}=0) - \prod_{j=1,\ldots,\ell}(1-p_{i_j})|.$$

Using this, we can show that Δ is bounded by ϵ^2 plus the sum of $\Delta_S(T)$ for $k^{O(\log(\frac{1}{\epsilon}))}$ many canonical rectangles T, each of dimension at most $2\log(\frac{1}{\epsilon})$. To bound $\Delta_S(T)$, note that the sampled point lies in T exactly when there is a specific setting of the most significant bits of $O(\log(\frac{1}{\epsilon}))$ of the Y_j's. Since the variables Y_j are defined by sampling from a binary (k',ϵ')-approximate space, if $k' = c'\cdot\log(\frac{1}{\epsilon})\log(\frac{4k}{\epsilon})$, for some constant c', then $\Delta_S(T)\le\epsilon'$. Fixing ϵ' very small yields $\Delta\le\epsilon$.

Theorem 5.2 *Assume $\frac{4k}{\epsilon}$ is a power of 2 and set $\ell=\log(\frac{4k}{\epsilon})$. Let the random variables $Y_1,\ldots,Y_n\in[0,\frac{4k}{\epsilon}-1]^n$ be defined by sampling $n\ell$ random bits $\{x_{i,j}: 1\le i\le n,\ 1\le j\le\ell\}$ from a binary (k',ϵ')–approximate distribution and by setting Y_i to be the number with binary representation $x_{i,1}x_{i,2}\ldots x_{i,\ell}$. If $k'=a_1\log(\frac{1}{\epsilon})\log(\frac{4k}{\epsilon})$, and $\epsilon'=\frac{\epsilon}{(2k)^{a_2\log(\frac{1}{\epsilon})}}$ for suitable constants a_1 and a_2, then the resulting sample S of all such Y's is (k,ϵ)-approximate.*

Using the binary (k',ϵ') approximate constructions of [NN90,AGHP92,ABN+92], we get a general (k,ϵ)-approximate sample space of size poly$(\log n, 1/\epsilon, k^{\log(\frac{1}{\epsilon})})$, if $k\ge a\log(\frac{1}{\epsilon})$ for some $a>0$.

5.1.2 A Bootstrapping construction

A second construction comes from bootstrapping the first construction in [EGL+92] with ideas from their second construction. Their first construction yields a sample space whose size is polynomial in $\frac{1}{\epsilon}$ but only polynomial in 2^k. To exploit this, we use ideas from [EGL+92] to reduce the problem of constructing a (k,ϵ)-approximate space to that of constructing a (k', ϵ')-approximate space where $k' \ll k$ with ϵ' not much smaller than ϵ.

Lemma 5.2 *If S is a (k', ϵ') approximate sample space, where $k' = \log(\frac{1}{\epsilon})$ and $\epsilon' = \epsilon^2/(\min(2^k, k^{\log(\frac{1}{\epsilon})}))$, then S is a (k, ϵ)-approximate sample space.*

We now get a (k, ϵ)–approximate sample space of size poly$(\log n, \frac{1}{\epsilon}, \min(2^k, k^{\log(\frac{1}{\epsilon})}))$ from the first construction of [EGL+92]. Though this proof is easier, we believe that the previous method can yield better results in special cases.

5.2 Sample Spaces Uniform on Neighborhoods

Derandomization has proved to be an essential tool over the last few years in obtaining deterministic algorithms both in the sequential and parallel domain. For several problems the only deterministic algorithms have been obtained by derandomization. As we have seen a very useful technique in derandomization is the use of *limited independence* of random variables. In this section we will be concerned only with random bits *i.e.* uniform binary random variables. Efficient constructions of sample spaces $S \subseteq \{0,1\}^n$ such that the distribution induced on $\{0,1\}^n$, by sampling uniformly from S, is k–wise independent are given in [Lub86,Jof74,ABI86]. These constructions have been used to design several deterministic algorithms from their randomized counterparts such as the NC algorithms for maximal independent set and other problems[Lub86,ABI86].

However, it can be shown that if $S \subseteq \{0,1\}^n$ is a sample space with an induced k–wise independent distribution then S must contain at least $(\binom{n}{1} + \ldots + \binom{n}{\lfloor k/2 \rfloor})$ points[ABI86, ASE92,CGH+85]. Thus, polynomial size constructions are possible only for constant k. Schulman[Sch92] observes that for several algorithms we do not need independence in all the $\binom{n}{k}$ subsets of indices and constructs sample spaces which are independent only for specified index sets of k elements. A *neighborhood* is a set of indices for which the independence constraint is to be satisfied. The main result in [Sch92] is

Theorem 5.3 *Given neighborhoods $N_1, \ldots, N_m \subseteq \{1, \ldots, n\}$ each of size at most k with each $i \in \{1, \ldots, n\}$ occurring in at most d neighborhoods, there exists an explicitly constructible sample space of size $O(d2^k)$ with the induced distribution independent on each of the specified neighborhoods.*

We are interested in the problem of establishing tight bounds on the size of the sample spaces independent on given neighborhoods.

Definition 9 *$S(n, k, m, d)$ denotes the size of the smallest sample space on n variables independent on m specified neighborhoods of size at most k, with each index occurring in at most d neighborhoods.*

In our notation, Theorem 5.3 says that $S(n, m, k, d) = O(d2^k)$. Consider the case when $k = 2$. Here, the input can be specified as a graph on n vertices where an edge $\{i, j\}$ implies that independence is to be satisfied over the set $\{i, j\}$. For $k = 2$, Theorem 5.3 gives a sample space of size $O(\Delta)$ where Δ is the maximum degree of the input graph. This is not optimal as shown by the following.

Lemma 5.3 *$S(n, 2, m, d) = O(\chi(G))$, where $\chi(G)$ is the chromatic number of the input graph G.*

PROOF. Fix color classes $C_1, \ldots, C_{\chi(G)}$ in G. Then construct sample space S' which is pairwise independent over all pairs on $\chi(G)$ new variables $Y_1, \ldots, Y_{\chi(G)}$. Sample space

S is defined by setting X_j equal to Y_k where the color class of the vertex j is C_k. Whenever vertices i and j are adjacent they belong to different color classes and hence the corresponding variables X_i and X_j are pairwise independent. We can explicitly construct such an S' with $|S'| = |S| = O(\chi(G))$[ABI86,Jof74,Lub86]. $\square$

If G is a bipartite graph the above sample space is of constant size where as the construction in [Sch92] has size $O(\Delta)$. Thus, for $k = 2$, Schulman's constructions essentially approximate the chromatic number by the maximum degree. Since G can be colored in NC with $\Delta + 1$ colors[Lub88] we get

Corollary 5.1 *There exists an explicitly constructible (in NC) sample space of size $O(\Delta)$.*

This is smaller than the NC constructions of[Sch92] which are of size $O(|E|)$.

However, $\chi(G)$ is not a lower bound as shown by the following class of graphs: Let $G(n) = (V, E)$, where $V = \{w \in \{-1, 1\}^n | \ w \text{ has } \frac{n}{2} \text{ ones } \}$ and with an edge between u and v if the dot product $u^T v$, over $\Re$, is 0. The following is a direct corollary of a result of [FW81].

Theorem 5.4 *If $n = 4p^k$, where p is an odd prime, then all independent sets in $G(n)$ are of size at most $2\binom{n-1}{\frac{n}{4}-1}$ and hence $\chi(G(n))$ is exponential in n.*

However there is a sample space, S, of size n, which satisfies all the independence constraints specified by $G(n)$: for $v \in V$, let $X_v = \frac{(1-v_i)}{2}$ at the i^{th} point in S, where v_i is the i^{th} component of v. By definition, vertices u and v are adjacent in $G(n)$ iff there are exactly $\frac{n}{4}$ positions i, such that $u_i = v_i = 1$. Thus, when u and v are adjacent, if we uniformly sample from S, we have $\Pr[(X_u = 0) \wedge (X_v = 0)] = \frac{1}{4}$. Thus, S satisfies all the pairwise independence constraints specified by $G(n)$.

We now show that a lower bound for $S(n, 2, m, d)$ is provided by the following interesting function of Lovász [Lov79]. In 1979, Lovász introduced the function $\nu(G)$, in his study of the Shannon capacity of a graph:

Definition 10 *An orthonormal representation of a graph G on n vertices is a system of unit vectors $(u_1, \ldots, u_n)$, $u_i \in \Re^d$, such that if i and j are adjacent then u_i and u_j are orthogonal. The* value *of the representation is* $\min_c \max_{1 \leq i \leq n} \frac{1}{(c^T u_i)^2}$ *where c ranges over all unit vectors. $\nu(G)$ is defined to be the minimum value over all representations of G.*

Theorem 5.5 *$S(n, 2, m, d) \geq \nu(G)$, where G is the input graph of independence constraints.*

Proof. A sample space S which satisfies pairwise independence constraints given by a graph G yields an orthonormal representation for G: Let u_i be obtained by normalizing the vector whose j component is -1 if X_i is 1 at the j^{th} point of S and 1 otherwise. For any two vertices i and j, the dot product $u_i^T u_j$ is proportional to the number of components at which they have the same value minus the number of components at which they disagree. If i and j are adjacent, the corresponding variables X_i and X_j are pairwise independent and hence $u_i^T u_j = 0$. Thus, $(u_1, \ldots, u_n)$ is an orthonormal representation for G with each vector of dimension $|S|$. The lower bound follows from the fact that if G admits a dimension d orthonormal representation then $\nu(G) \leq d$ [Lov79]. □

Now consider the case $k > 2$. Let $N_1, \ldots, N_m$ be the explicitly given neighborhoods over which we want independence. Construct a graph $G_{n,k} = (V, E)$ with vertex set

$V = \{I \subseteq \{X_1, \ldots, X_n\} \mid |I| \leq k\}$ and the edge set $E = \{(I, I') \mid I \cup I' \subseteq N_j$ for some $j\}$. Now, any sample space S which satisfies independence on the neighborhoods N_j, yields a sample space of the same size which satisfies the pairwise constraints of $G_{n,k}$: for $I \subseteq \{X_1, \ldots, X_n\}$ define $Y_I = \oplus_{i \in I} X_i$. Since S satisfies all the specified independence constraints, by the definition of k-wise independence, the Y_I's satisfy the pairwise constraints of $G_{n,k}$. Thus a lower bound for $S(n,k,m,d)$ is $\nu(G_{n,k})$. This generalizes the lower bounds of [ABI86,ASE92,CGH$^+$85] since if we need independence in all k sized neighborhoods, then $G_{n,k}$ has a clique of size $\binom{n}{1} + \ldots + \binom{n}{\lfloor k/2 \rfloor}$ and for all graphs G, $\nu(G) \geq \omega(G)$ [Lov79], where $\omega(G)$ is the clique number of G. The main question we address in this chapter is whether the converse is true *i.e.*

> Can we define n functions on the variables associated with $G_{n,k}$ such that given any sample space S satisfying the constraints of $G_{n,k}$ the induced distribution on the n new variables satisfies the given $(k+1)$–wise independence constraints?

Note that if n is small then we can easily answer the question in the negative. When n is large, however, there will be enough entropy in the system, so that such a construction is conceivable.

5.2.1 Independence can not be amplified

In this section we consider *logical consequences* of pairwise independence of n binary random bits and, specifically, whether any fixed functions of pairwise independent bits could yield higher independence.

Definition 11 *An event E occurring with probability p is a* logical consequence *of pairwise independence of random bits $X_1, \ldots, X_n$ if for every distribution $\mathcal{D}$ which is pairwise independent on the X_i's the event E occurs with probability p.*

We show that there is no event E such that E occurs with probability $\frac{1}{8}$ is a logical consequence of pairwise independence of n bits, for all n. This implies that there are no fixed functions on n pairwise–independent bits which are 3-wise independent, thus answering the question in the end of the previous section in the negative. In fact we shall show that a single sample space exists which satisfies all the pairwise independence constraints but diagonalizes against all fixed functions of n variables. The proof technique is novel as it uses a p-adic reduction as well as volume and dimension estimates in a certain high dimensional space. Our proof technique also directly extends to show that no fixed functions can amplify k–wise independence into $(k+1)$–wise independence.

Theorem 5.6 *Let $X_1, \ldots, X_n$ be uniform binary random variables. There exists a sample space $S \subseteq \{0,1\}^n$, such that, the induced distribution on the X_i's is unbiased and* pairwise independent*, but for any fixed functions $Y_j = f_j(X_1, \ldots, X_n)$, $1 \le j \le 3$, the random variables Y_1, Y_2 and Y_3 are not 3–wise independent unbiased random bits.*

Proof. In the proof, elements of $\{0,1\}^n$ are viewed as subsets of $\{1, \ldots, n\}$ where an n–bit vector is treated as the characteristic vector of the subset it represents. A pairwise–independent distribution must satisfy three sets of constraints: $\sum_\alpha p_\alpha = 1$, $\sum_{i \in \alpha} p_\alpha = \frac{1}{2}$ for $1 \le i \le n$, and $\sum_{i,j \in \alpha} p_\alpha = \frac{1}{4}$ for $1 \le i \ne j \le n$, where p_α is the probability assigned to point $\alpha \in \{0,1\}^n$. Conversely, any assignment p_α, $\alpha \in \{0,1\}^n$, which satisfies these constraints along with the condition that $p_\alpha \ge 0$ for all α, defines a distribution which is pairwise independent. We show that if event E occurring with probability p is a logical consequence of pairwise independence then p must be a integer combination of $1, \frac{1}{2}$, and $\frac{1}{4}$.

We can rewrite the above constraints as the following linear system of equations $\sum_{\alpha\in\{0,1\}^n} a_{S,\alpha}p_\alpha = b_S$ for $S = \emptyset, \{1\}, \ldots, \{n\}, \{1,2\}, \{1,3\}, \ldots, \{n-1,n\}$ where

$$a_{S,\alpha} = \begin{cases} 1 & \text{if } S \subseteq \alpha \\ 0 & \text{otherwise} \end{cases}$$

Let A be the matrix of coefficients of this linear system. We index the rows of A by the subset S that defines the row. Thus a pairwise independent distribution must satisfy the linear system $Ap = b$. First, we observe that the matrix A is of full row rank over all fields. To see this, consider columns of A defined by the elements of $\{0,1\}^n$ corresponding to the following subsets of $\{1,\ldots,n\}$: $\emptyset, \{1\}, \ldots, \{n\}, \{1,2\}, \ldots, \{1,n\}$ $\{2,3\}, \ldots, \{n-1,n\}$. The resulting sub–matrix is a square matrix which is upper triangular with 1's on the diagonal and hence invertible over all fields. Thus, A is of rank $1 + \binom{n}{1} + \binom{n}{2}$ over all fields.

Assume that there exist three functions f_1, f_2 and f_3, such that for any distribution which satisfies the pairwise–independence constraints for the X_i's , the random variables $Y_j = f_j(X_1, \ldots, X_n)$, $1 \leq j \leq 3$ are independent and unbiased random bits. Consider the event $E = [(Y_1 = 1) \wedge (Y_2 = 1) \wedge (Y_3 = 1)]$ which occurs with probability $\frac{1}{8}$ if the Y_j's are independent. Construct the 2^n dimensional $\{0,1\}$ vector $\vec{v}$ with $\vec{v}_\alpha = 1$ if E occurs at α. By assumption, any nonnegative solution p of the system $Ap = b$, also satisfies $\vec{v}p = \frac{1}{8}$ since p defines a pairwise independent distribution. We claim that this can happen only if $\vec{v}$ is in the span of the rows of A. To see this, first fix p^* to be the vector with each entry $\frac{1}{2^n}$, which is a nonnegative solution of the system $Ap = b$ as it corresponds to the unbiased n–wise independent distribution. Let $z_1, \ldots, z_d$ be a basis for the null space of A, where $d = 2^n - 1 - \binom{n}{1} - \binom{n}{2}$. All solutions to the linear system $Az = b$ are of the form $p^* + \sum_{i=1}^{d} x_i z_i$, where x_i ranges over the underlying field Q. For small x_i the point $p^* + x_i z_i$ is in $(0,1)^{2^n}$ as the unit cube $(0,1)^{2^n}$ is an open set. Therefore $p^* + x_i z_i$ defines a probability distribution which satisfies $Ap = b$ and hence is a pair–wise independent distribution and so, by assumption, $v(p^* + x_i z_i) = \frac{1}{8}$. Since $vp^* = \frac{1}{8}$ as well, $v \cdot z_i = 0$ for all i and thus, v is in the linear span of the rows of A. Therefore we can write $\vec{v}$ as a linear combination of the rows, $A_{\emptyset}, \ldots, A_{\{n-1,n\}}$, of A with rational coefficients. Multiplying by the least common multiple of the denominators, we have

$$c\vec{v} = \sum_{S} c_S A_S$$

where the c_S's and c are integers with $gcd(c, c_{\emptyset}, \ldots c_{\{n-1,n\}}) = 1$. We claim that $c = 1$. If not then consider a prime q which occurs in c and reduce the above equation modulo q. We can derive that the rows of A are linearly dependent over Z_q which contradicts the fact that the matrix A has full row rank over all fields. Therefore $\vec{v}$ is an integer combination of the rows of A. By assumption, if $Ap = b$ with p nonnegative then we have $\vec{v}p = \frac{1}{8}$ and since $\vec{v}$ is an integer combination of the rows of A, this implies that $\frac{1}{8}$ is an integer combination of $1, \frac{1}{2}$, and $\frac{1}{4}$ which is a contradiction.

We showed that the (affine) hyperplane $v \cdot p = \frac{1}{8}$ does not contain the portion of the (affine) d–dimensional solution space $Ap = b$ inside the unit cube $(0,1)^{2^n}$. Thus the intersection of the hyperplane with the d-dimensional solution space $Ap = b$ is a $(d-1)$–dimensional affine space. Excluding such points on the lower dimensional space of the intersection, we can find points in the unit cube $(0,1)^{2^n}$ which satisfy the pairwise independence constraints and yet the variables Y_1, Y_2, Y_3 are not independent. Note that the normal vector v to the hyperplane $vp = \frac{1}{8}$ was defined by the event $[(Y_1 = 1) \wedge (Y_2 = 1) \wedge (Y_3 = 1)]$ and there are only finitely many such v corresponding to different choices for f_1, f_2, and f_3. Thus, in fact, we can find a pair–wise independent distribution p which is on none of the finitely many hyperplanes $v \cdot p = \frac{1}{8}$, and thus for no choice of the functions f_1, f_2 and f_3 can the variables Y_1, Y_2, Y_3 can be independent and unbiased random bits. □

What if we do not require the variables Y_1, Y_2, Y_3 to be necessarily unbiased? In fact the above proof shows that as long as none of Y_1, Y_2, Y_3 is a trivial random variable, i.e., a constant 0 or 1 valued random variable, the same conclusion still follows. (For trivial random variables, the conclusion is certainly false, as they are always "independent").

To see this, notice that $\sum_{i_1,i_2,i_3=0,1} \Pr[(Y_1 = i_1) \wedge (Y_2 = i_2) \wedge (Y_3 = i_3)] = 1$. Thus, for some i_1, i_2, i_3, $\Pr[(Y_1 = i_1) \wedge (Y_2 = i_2) \wedge (Y_3 = i_3)] \leq \frac{1}{8}$. But it can't equal 0, otherwise, by independence, we would have $\Pr[Y_j = i_j] = 0$ for some j, contrary to our hypothesis. Now the same proof goes through, since no positive quantity less than or equal to $\frac{1}{8}$ can be an integer combination of $1, \frac{1}{2}$ and $\frac{1}{4}$.

Corollary 5.2 *Let $X_1, \ldots, X_n$ be uniform binary random variables. Then there is a sample space $S \subseteq \{0,1\}^n$, such that, the induced distribution on the X_i's is unbiased and* pairwise independent, *but for any functions $Y_j = f_j(X_1, \ldots, X_n)$, $1 \leq j \leq 3$, the variables Y_1, Y_2 and Y_3 are not 3–wise independent, unless one of the Y_j's is constant.*

The proof of the theorem and the corollary directly generalize to the case $k > 2$.

Theorem 5.7 *Let $X_1, \ldots, X_n$ be uniform binary random variables. There exists a sample space $S \subseteq \{0,1\}^n$, such that, the induced distribution on the X_i's is unbiased and k–wise independent, but for any fixed functions $Y_j = f_j(X_1, \ldots, X_n)$, $1 \leq j \leq (k+1)$, the random variables $Y_1, \ldots, Y_{k+1}$ are not $(k+1)$–wise independent unless one of the Y_j's is a constant.*

5.2.2 Enumerating all logical consequences

We turn to a slightly general problem: Let $X_1, \ldots, X_n$ be binary random variables with $\Pr[X_i = 1] = p_i$. We are interested in enumerating all possible logical consequences of pairwise independence of these variables. From the proof of Theorem 5.6, the characteristic vector of any event E that is a logical consequence must be an integer combination of the rows of the constraint matrix A. Furthermore, not all integer combinations are allowed: the resulting vector, which is the characteristic vector of E, must have $0, 1$ entries. This additional constraint allows us to enumerate all possibilities.

Theorem 5.8 *The only logical consequences of pairwise independence of n binary random variables $X_1, \ldots, X_n$ with $\Pr[X_i = 1] = p_i$, are events with probabilities 0, p_i, p_ip_j, $p_i(1-p_j)$, $(1-p_i)(1-p_j)$, $p_i(1-p_j)+p_jp_k$, $p_i(1-p_j)+p_j(1-p_i)$, $p_i+p_j+p_k-p_ip_j-p_ip_k-p_jp_k$, $p_i+p_j-p_ip_j-p_jp_k$, $p_i+p_j-p_ip_j-p_ip_k-p_lp_j+p_lp_k$ and $p_ip_j+p_k(1-p_j)+(1-p_k)(1-p_i)$ and their complements, where i, j, k and l are distinct indices.*

Proof. As in the previous section we index the rows of the constraint matrix A by the subset which defines the row of the matrix. In any integer combination of the rows of A, we denote the coefficient of the row $A_\emptyset$ as c_0, the coefficient of the row $A_{\{i\}}$ as

c_i, $1 \leq i \leq n$ and the coefficient corresponding to $A_{\{i,j\}}$ as c_{ij}, $1 \leq i \neq j \leq n$. Also, elements of $\{0,1\}^n$ are viewed as subsets of $\{1,\ldots,n\}$ and we fix some arbitrary ordering of these subsets and use it to index 2^n dimensional vectors. Thus, for a 2^n dimensional vector $\vec{u}$, $\vec{u}_S$ denotes the entry of $\vec{u}$ corresponding to subset S.

Consider an event E whose characteristic vector, $\vec{v}$, is an integer combination of the rows of A. The condition that the integer combination result in a 0,1–vector forces restrictions on the coefficients which we outline below. We repeatedly use the fact that for any $S \subseteq \{1,\ldots,n\}$, $\vec{v}_S = c_0 + \sum_{i \in S} c_i + \sum_{i<j \in S} c_{ij}$. Assume, without loss of generality, that $\vec{v}_\emptyset = 0$, for if not, we can consider the complement of the event E. Thus, $c_0 = 0$ and $\vec{v}_{\{i\}} = c_i$ for all i and hence, c_i must be 0 or 1.

Claim 5.1 *At most three of the coefficients c_i, $1 \leq i \leq n$ can be 1.*

Proof. Since $\vec{v}_{\{i_1,i_2\}} = c_{i_1} + c_{i_2} + c_{i_1 i_2}$, if both c_{i_1} and c_{i_2} are 1 then $c_{i_1 i_2}$ must be -1 or -2. If four different coefficients c_{i_j}, $1 \leq j \leq 4$, are 1 then $\vec{v}_{\{i_1,i_2,i_3,i_4\}} = \sum_j c_{i_j} + \sum_{j<k} c_{i_j i_k} \leq -2$. □

If c_i and c_j are both 0 then since $\vec{v}_{\{i,j\}} = c_i + c_j + c_{ij}$, the coefficient c_{ij} must be 0 or 1.

Claim 5.2 *There is at most one pair $\{i,j\}, i \neq j$ such that $c_i = c_j = 0$ and $c_{ij} = 1$.*

Proof. Assume that there are 2 such pairs $\{i_1,i_2\}$ and $\{i_3,i_4\}$. Since some of the indices may be equal the set $S = \{i_1,i_2\} \cup \{i_3,i_4\}$ may contain either 3 or 4 elements. By definition $\vec{v}_S = \sum_{i_k \in S} c_{i_k} + \sum_{i_k < i_l \in S} c_{i_k i_l}$. For any pair $\{i_k, i_l\}$ in S, $i_k \neq i_l$, since $c_{i_k} = c_{i_l} = 0$ the coefficient $c_{i_k i_l}$ is either 0 or 1 and there are at least 2 different pairs with this coefficient 1. Thus, $\vec{v}_S \geq 2$ which is a contradiction and hence there can be at most one such pair. □

If $c_i = 1$ and $c_j = 0$, the coefficient c_{ij} should be either 0 or -1. In the next two claims we establish properties of such pairs.

Claim 5.3 *There are at most 2 pairs $\{i, j\}, i \neq j$ such that $c_i = 1$, $c_j = 0$ and $c_{ij} = -1$. If there are 2 such pairs $\{i_1, j_1\}$ and $\{i_2, j_2\}$ then $j_1 \neq j_2$ and $c_{j_1 j_2} = 1$.*

PROOF. Assume that there are three pairs $\{i_1, j_1\}$,$\{i_2, j_2\}$ and $\{i_3, j_3\}$ satisfying the above conditions. To handle the case of some of the coefficients being the same we first consider the set $I = \{i_1\} \cup \{i_2\} \cup \{i_3\}$. If $i_k, i_l \in I$ with $i_k \neq i_l$ then, since c_{i_k}=c_{i_l}=1, the coefficient $c_{i_k i_l}$ is either -1 or -2. Similarly consider the set $J = \{j_1\} \cup \{j_2\} \cup \{j_3\}$. For any two indices $j_k, j_l \in J$, $j_k \neq j_l$, the coefficient $c_{i_k i_l}$ is either 0 or 1. However by Claim 5.2 at most one such coefficient can be 1. If $i_k \in I$ and $j_l \in J$ then the coefficient $c_{i_k j_l}$ is either 0 or -1 and by assumption there are at least three such pairs with coefficient -1. Finally, let $S = I \cup J$. From the previous statements we can deduce that $\vec{v}_S \leq |I| - \binom{|I|}{2} + 1 - 3 \leq -1$ which is a contradiction.

Assume that 2 pairs $\{i_1, j_1\}$ and $\{i_2, j_2\}$ exist that satisfy the conditions of the claim. Let $S = \{i_1, j_1\} \cup \{i_2, j_2\}$. If $j_1 = j_2$ then $i_1 \neq i_2$ and since $c_{i_1 i_2} \leq -1$, we have $\vec{v}_S \leq -1$. Thus $j_1 \neq j_2$. Again if $c_{j_1 j_2} = 0$ we can show that $\vec{v}_S < 0$ irrespective of whether $i_1 = i_2$ or not. □

Claim 5.4 *If $c_i = 1$ and $c_{jk} = 1$ then either $c_{ij} = -1$ or $c_{ik} = -1$.*

PROOF. Firstly, since c_j and c_k are both nonnegative if $c_{jk} = 1$ then both c_j and c_k must be 0. Hence, if both the coefficients c_{i1} and c_{ik} are 0, then $\vec{v}_{\{i,j,k\}} \geq 2$. □

From Claim 5.1 at most three of the c_i's are 1. Let N denote the number of c_i's which are one. We consider each case separately.

N=0: By Claim 5.2 the event E must have probability either 0 or $p_i p_j$ for some $i < j \leq n$.

N=1: Let $c_i = 1$. Then by Claim 5.3 at most 2 of the coefficients c_{ij} can be -1. If there are 2 such coefficients $c_{ij} = c_{ik} = -1$ then $c_{jk} = 1$. The other claims now imply that no other coefficient can be non zero and the event E has probability $p_i - p_i p_j - p_i p_k + p_j p_k$ which is $1 - [p_i p_j + (1-p_j)p_k + (1-p_i)(1-p_k)]$. If exactly one $c_{ij} = -1$ then by Claims 5.2 and 5.4 the resulting events must have probabilities $p_i - p_i p_j$, $p_i - p_i p_j + p_j p_k$. If no $c_{ij} = -1$ then by Claim 5.4 the event E has probability p_i.

N=2: Let $c_i = c_j = 1$ with $i \neq j$. Then the coefficient c_{ij} must be either -1 or -2. If $c_{ij} = -2$, we argue that all other coefficients must be 0. If either $c_{ik} = -1$ or c_{jk} is -1 for some $k =\neq i, j$, then we get a contradiction since then $\vec{v}_{\{i,j,k\}} = -1$. Thus, by Claim 5.4 all other coefficients are 0 and the event has probability $p_i + p_j - 2p_i p_j = p_i(1-p_j) + p_j(1-p_i)$.

Let $c_{ij} = -1$. If $c_{kl} = 1$ for some pair k, l with $c_k = c_l = 0$, then the event has probability $p_i + p_j - p_i p_j - p_i p_k - p_l p_j + p_l p_k$, since by Claims 5.2, 5.3 and 5.4 all other coefficients must be 0. If $c_{kl} = 0$ for all k, l with $c_k = c_l = 0$, then by Claim 5.3 the event must have probability either $p_i + p_j - p_i p_j$, which is $1 - [(1-p_i)(1-p_j)]$, or $p_i + p_j - p_i p_j - p_i p_k$.

N=3: Let $c_i = c_j = c_k = 1$. In this case it can be seen that the cross coefficients $c_{ij} = c_{ik} = c_{jk} = -1$. Claims 5.4 and 5.3 imply that no coefficient $c_{j_1 j_2}$ can be nonzero, if $c_{j_1} = c_{j_2} = 0$. If any of the coefficients c_{il}, c_{jl} or c_{kl} is -1, for some $l \notin \{i, j, k\}$, then $\vec{v}\{i, j, k, l\} = -1$. Hence, all other coefficients are 0 and the event has probability $p_i + p_j + p_k - p_i p_j - p_j p_k - p_k p_i$. $\square$

Note that we can define events which actually realize these possible probabilities. Let P, Q, R, S denote the pair-wise independent events $X_i = 1$, $X_j = 1$, $X_k = 1$, and $X_l = 1$, respectively. Then the listed probabilities in Theorem 5.8 are realized by the following events: $P\overline{P}$, P, PQ, $P\overline{Q}$, $\overline{P} \wedge \overline{Q}$, $(\overline{Q}P) \vee (QR)$, $(P\overline{Q}) \vee (Q\overline{P})$, $(P \wedge \overline{Q}) \vee (Q \wedge \overline{R}) \vee (R \wedge \overline{P})$, $(\overline{Q}P) \vee (Q\overline{R})$, $P[(RS\overline{Q}) \vee (\overline{R} \wedge \overline{(QS)})] \vee \overline{P}[SR \vee \overline{S}Q]$, $PQ \vee \overline{Q}R \vee (\overline{R} \wedge \overline{P})$.

If all the probabilities are equal to p then the only logical consequences have probabilities $0, p, p^2, p(1-p), 2p(1-p), 3p(1-p), (1-p)^2$ or their complements.

5.3 Discussion

In this chapter we gave several constructions of sample spaces which satisfied required constraints on the induced distribution. These constructions are more efficient than previously known constructions and they have obvious applications to randomness efficient computation and derandomization. We also considered the problem of lower bounds for the size of such sample spaces and here again the new lower bounds we show are better than previously known ones.

In the first problem we considered of sample spaces approximately independent for multivalued random variables the constructions we give, though better than previously known ones, does not still yield efficient deterministic algorithms. Probabilistically one can show the existence of a small sample spaces which yields deterministic algorithms. Thus an obvious open question is to construct explicitly such small sample spaces.

In the problem of sample spaces which are uniform on specified neighborhoods an open question is the problem of good upper bounds for the case of k–wise independence and a tight characterization for the case of $k = 2$. We conjecture that the optimal sample space for the case of $k = 2$ is the Lovasz ν function.

Bibliography

[ABI86] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[ABN$^+$92] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of asymptotically good, low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38:509–516, 1992.

[Adl78] L. Adleman. Two theorems on random polynomial time. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k–wise independent random variables. *Random Structures and Algorithms*, 3(3):289–303, 1992.

[All86] E. Allender. The complexity of sparse sets in p. In *Proceedings of the 1$^{\text{st}}$ Structure in Complexity Theory Conference*, pages 1–10. Springer–Verlag, Lecture Notes in Computer Science #223, 1986.

[ALMS92] S. Arora, C. Lund, R. Motwani, and M. Sudan, M. amd Szegedy. Proof verification and the hardness of approximation problems. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[AM77] L. Adleman and K. Manders. Reducibility, randomness and intractibility. In *Symposium on Theory of Computing*, pages 151–163, 1977.

[AMN90] Y. Azar, R. Motwani, and J. Naor. Approximating arbitrary probability distributions using small sample spaces. Manuscript, 1990.

[AN92] N. Alon and M. Naor. Derandomization, witnessses for Boolean matrix multiplication and construction of perfect hash functions. Manuscript, 1992.

[ASE92] N. Alon, J. Spencer, and P. Erdős. *The Probabilistic Method.* Wiley–Interscience Series, John Wiley & Sons, Inc., New York, 1992.

[BCP83] A. Borodin, S. A. Cook, and N. Pippenger. Parallel computation for well–endowed rings and space–bounded probabilistic machines. *Information and Control*, 58:113–136, 1983.

[Bei91] R. Beigel. Relativized counting classes: Relations among thresholds, parity, mods. *Journal of Computer and System Sciences*, 42, 1991.

[Ber90] B. Berger. *Using Randomness to design efficient deterministic algorithms.* Ph.D. dissertation, Massachussets Institute of Technology, 1990.

[BG82] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 55(1–3):80–88, 1982.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BoGKW88] M. Ben-or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi prover interactive proofs: How to remove intractability. In *Proceedings of the 20^{th} Annual ACM Symposium on Theory of Computing*, pages 113–131, 1988.

[BR91] B. Berger and J. Rompel. Simulating $(\log^c n)$-wise independence in NC. *J. Assoc. Comput. Mach.*, 38(4):1026–1046, 1991.

[BRS89] B. Berger, J. Rompel, and P. W. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 54–59, 1989.

[BS90] B. Berger and P. W. Shor. Approximation algorithms for the maximum acyclic subgraph problem. In *Proc. ACM/SIAM Symposium on Discrete Algorithms*, pages 236–243, 1990.

[BW80] J. L. Bentley and D. Wood. An optimal worst–case algorithm for reporting intersections of rectangles. *IEEE Trans. Comput.*, 29:571–577, 1980.

[CC94] J–Y. Cai and S. Chari. A note on the independence of random variables. Technical Report 94–05, State University of New York at Buffalo, 1994.

[CF90] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.

[CG89] B. Chor and O. Goldreich. On the power of two–point sampling. *Journal of Complexity*, 5:96–106, 1989.

[CGH+85] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem or t–resilient functions. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 396–407, 1985.

[CGM92] P. M. Camerini, G. Galbiati, and F. Maffioli. Random pseudopolynomial algorithms for exact matroid problems. *Journal of Algorithms*, 13(2):258–273, 1992.

[CH90] J-Y. Cai and L. Hemachandra. On the Power of Parity Polynomial Time. *Mathematical Systems Theory*, 23(2):95–106, 1990.

[CM93] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. In *Proc. ACM/SIAM Symposium on Discrete Algorithms*, pages 281–290, 1993.

[CRS93] S. Chari, P. Rohatgi, and A. Srinivasan. Randomness efficient unique element isolation with application to perfect matching and related problems. In *Proc. ACM Symposium on Theory of Computing*, pages 458–467, 1993.

[CRS94] S. Chari, P. Rohatgi, and A. Srinivasan. Improved algorithms via approximations of probability distributions. In *Proc. ACM Symposium on Theory of Computing*, 1994.

[Csa76] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM J. Comput.*, 5:618–623, 1976.

[dB51] N. de Bruijn. On the number of positive integers $\leq x$ and free of prime factors $> y$. *Proc. Kon. Ned. Akad. Wet.(Indag. Math. 13)*, A54:50–60, 1951.

[DGMP92] M. Dietzfelbinger, J. Gil, Y. Matias, and N. Pippenger. Polynomial hash functions are reliable. In *Automata, Languages, and Programming (ICALP)*, pages 235–246. Springer–Verlag Lecture Notes in Computer Science #623, 1992.

[DK86] E. Dahlhaus and M. Karpinski. The matching problem for strongly chordal graphs is in NC. Technical Report 855–CS, University of Bonn, 1986.

[EGL+92] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković. Approximations of general independent distributions. In *Proc. ACM Symposium on Theory of Computing*, pages 10–16, 1992.

[EK68] P. Erdős and D. Kleitman. On coloring graphs to maximize the proportion of multicolored k-edges. *Journal of Combinatorial Theory*, 5(2):164–169, 1968.

[FK92] M. Fellows and N. Koblitz. Self–Witnessing Polynomial Time Complexity and Prime Factorization. In *Proceedings of the* 7^{th} *Annual Structure in Complexity Theory Conference*, pages 107–111, 1992.

[FW81] P. Frankl and R. Wilson. Intersection theorems with geometric consequences. *Combinatorica*, 1:357–368, 1981.

[GK87] D. Yu. Grigoriev and M. Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 166–172, 1987. See also D. Yu. Grigoriev, M. Karpinski and M. F. Singer, Fast parallel algorithms for multivariate polynomial interpolation over finite fields, *SIAM J. Comput.*, Vol. 19, 1990, 1059–1063.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GPST92] A. V. Goldberg, S. A. Plotkin, D. B. Shmoys, and É. Tardos. Using interior–point methods for fast parallel algorithms for bipartite matching and related problems. *SIAM J. Comput.*, 21(1):140–150, 1992.

[GPV93] A. V. Goldberg, S. A. Plotkin, and P. M. Vaidya. Sublinear–time parallel algorithms for matching and related problems. *Journal of Algorithms*, 14(2):180–213, 1993.

[Gro92] L. K. Grover. Fast parallel algorithms for bipartite matching. In *Proceedings of the Integer Programming and Combinatorial Optimization Conference*, pages 367–384, 1992.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[Jof74] A. Joffe. On a set of almost deterministic k–independent random variables. *The Annals of Probability*, 2(1):161–162, 1974.

[Kar86] H. J. Karloff. A Las Vegas RNC algorithm for maximum matching. *Combinatorica*, 6(4):387–391, 1986.

[Kar91] R. M. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, 34:165–201, 1991.

[Kas67] P. W. Kastelyn. Graph theory and crystal physics. In F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, New York, 1967.

[Kel92] P. Kelsen. On the parallel complexity of computing a maximal independent set in a hypergraph. In *Proc. ACM Symposium on Theory of Computing*, pages 339–350, 1992.

[Knu82] D. E. Knuth. *The art of computer programming*, volume 1, pages 68–70. Addison–Wesley, 1982.

[KR88] H. J. Karloff and P. Raghavan. Randomized algorithms and pseudorandom numbers. In *Proc. ACM Symposium on Theory of Computing*, pages 310–321, 1988.

[KSTT92] J. Köbler, U. Schöning, S. Toda, and J. Toran. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44:272–286, 1992.

[KUW86] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in Random NC. *Combinatorica*, 6(1):35–48, 1986.

[KVV85] D. Kozen, U. V. Vazirani, and V. V. Vazirani. NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching. In *Proceedings, FST & TCS Conference*, Lecture Notes in Computer Science # 206, pages 496–503. Springer-Verlag, Berlin, 1985.

[KW85] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *J. Assoc. Comput. Mach.*, 32:762–773, 1985.

[Lit74] C. H. C. Little. An extension of Kastelyn's method of enumerating the 1–factors of planar graphs. In D. Holton, editor, *Combinatorial Mathematics, Proc. Second Australian Conference*, pages 63–72. Lecture Notes in Mathematics 403, Springer–Verlag, Berlin, 1974.

[Lov79] L. Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, IT–25(1):1–7, 1979.

[LP86] L. Lovász and M. Plummer. *Matching Theory.* North–Holland, Amsterdam, 1986.

[Lub86] M. Luby. A fast parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[Lub88] M. Luby. Removing randomness in parallel computation without a processor penalty. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 162–173, 1988.

[LY94] C. Lund and M. Yannnakakis. On the hardness of approximating minimization problems. In *Proc. ACM Symposium on Theory of Computing*, pages 286–293, 1994.

[MN89] G. L. Miller and J. Naor. Flow in planar graphs with multiple sources and sinks. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 112–117, 1989.

[MNN89] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 8–13, 1989.

[MV80] S. Micali and V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 17–27, 1980.

[MVV87] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[Nis90] N. Nisan. Pseudorandom generators for space–bounded computation. In *Proc. ACM Symposium on Theory of Computing*, pages 204–212, 1990.

[NN90] J. Naor and M. Naor. Small–bias probability spaces: efficient constructions and applications. In *Proc. ACM Symposium on Theory of Computing*, pages 213–223, 1990.

[NSV92] H. Narayanan, H. Saran, and V. V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arborescences and edge–disjoint spanning trees. In *Proc. ACM/SIAM Symposium on Discrete Algorithms*, pages 357–366, 1992.

[NSW92] N. Nisan, E. Szemerédi, and A. Wigderson. Undirected connectivity in $O(\log^{1.5} n)$ space. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 24–29, 1992.

[OH91] M. Ogiwara and L. Hemachandra. A Complexity theory for feasible closure properties. In *Proceedings of the Sixth Annual Conference on Structure in Complexity Theory*, pages 16–27, 1991.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer, New York, 1985.

[PS92] A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proc. ACM Symposium on Theory of Computing*, pages 581–592, 1992.

[PY82] C. H. Papadimitriou and M. Yannakakis. The complexity of restricted spanning tree problems. *J. Assoc. Comput. Mach.*, 29(2):285–309, 1982.

[PZ83] C. Papadimitriou and S. Zachos. Two Remarks on the Power of Counting. In Proc. 6^{th} GI Conference on Theoretical Computer Science,. *Lecture Notes in Computer Science Springer-Verlag, #145*, pages 276–296, 1983.

[Rag88] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[Roh94] P. Rohatgi. *On Properties of Random Reductions*. Ph.D. dissertation, Cornell University, 1994.

[Sch92] L. J. Schulman. Small spaces uniform on neighborhoods. In *Proc. ACM Symposium on Theory of Computing*, pages 17–25, 1992.

[Spe86] J. Spencer. Balancing vectors in the max norm. *Combinatorica*, 6:55–65, 1986.

[Spe87] J. Spencer. *Ten Lectures on the Probabilistic Method*. SIAM, Philadelphia, 1987.

[SSS93] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. In *Proc. ACM/SIAM Symposium on Discrete Algorithms*, pages 331–340, 1993. Expanded version available as Technical Report TR 92-1305, Department of Computer Science, Cornell University, October 1992.

[SV86] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33:75–87, 1986.

[Tar91] J. Tarui. Randomized polynomials, Threshold circuits and the Polynomial hierarchy. In *Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science #480, pages 238–250, 1991.

[Thr93] W. Thrash. A note on the least common multiples of dense sets of integers. Technical Report #93-02-04, Department of Computer Science & Engineering, University of Washington, Seattle, February 1993.

[Tiw87] P. Tiwari. Parallel algorithms for instances of linear matroid parity with small number of solutions. Technical Report RC 12766, IBM T.J.Watson Research Center, May 1987.

[Tod91] S. Toda. PP is as hard as the Polynomial-time Hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[Tom91] M. Tompa. Lecture notes on probabilistic algorithms and pseudorandom generators. Technical Report #91-07-05, Department of Computer Science & Engineering, University of Washington, Seattle, July 1991.

[Vai90] P. M. Vaidya. Reducing the parallel complexity of certain linear programming problems. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 583–589, 1990.

[Val82] L. G. Valiant. A scheme for fast parallel communication. *SIAM J. Comput.*, 11:350–361, 1982.

[Vaz89] V. V. Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$–free graphs and related problems. *Information and Computation*, 80:152–164, 1989.

[VB81] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proc. ACM Symposium on Theory of Computing*, pages 263–277, 1981.

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986.

[WC79] M.N. Wegman and J.L. Carter. New classes and applications of hash functions. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 175–182, 1979.

[Wei91] J. Wein. Las Vegas RNC algorithms for unary weighted perfect matching and T–join problems. *Information Processing Letters*, 40:161–167, 1991.

[Wel76] D. J. A. Welsh. *Matroid Theory.* Academic Press, New York, 1976.