Progress in Computational Complexity Theory

Jin-Yi Cai *
Computer Sciences Department
University of Wisconsin
Madison, WI 53706. USA.
Email: jyc@cs.wisc.edu
and
Tsinghua University
Beijing, China

Hong Zhu†
Computer Sciences Department,
Fudan University
Shanghai 200433, China.
Email: hzhu@fudan.edu.cn

Abstract

We briefly survey a number of important recent achievements in Theoretical Computer Science (TCS), especially Computational Complexity Theory. We will discuss the PCP Theorem, its implications to inapproximability on combinatorial optimization problems; space bounded computations, especially deterministic logspace algorithm for undirected graph connectivity problem; deterministic polynomial-time primality test; lattice complexity, worst-case to average-case reductions; pseudorandomness and extractor constructions; and Valiant's new theory of holographic algorithms and reductions.

Key words: Theoretical Computer Science, Computational Complexity Theory, PCP Theorem, Inapproximability, Logspace complexity, Reingold's Theorem, GAP problem, Primality testing, Complexity of lattice problems, Worst-case to average-case reductions, Pseudorandomness, Extractors, Holographic algorithms.

1 Background

The field of Theoretical Computer Science (TCS), especially Computational Complexity Theory, is arguably the most foundational aspect of Computer Science. It deals with fundamental questions such as, what is feasible computation, and what can and cannot be computed with a reasonable amount of computational resources in terms of time and/or space. It offers new and profound insight into concepts which have fascinated mankind for generations—proof and verification, randomness, efficiency, security, and of course computation.

This field is also one of the most vibrant. A great deal of fascinating research is being conducted with ever more sophisticated mathematical techniques. The results are deep, elegant and surprising. In this brief survey of some recent striking developments, we wish to sample a few of the highlights, to give the reader a flavor of this great achievement. We decided not to write something like an "executive survey" which only scratches the surface with some vague generalities. Instead we decided to delve deeper, to offer a glimpse of the true frontier as an active researcher sees it. And what a glorious frontier it is! We hope the earnest reader will come away with a new appreciation

*Supported by an NSF Grant, USA (CCR-0208013 and CCR-0511679).
†Supported by a grant from National Natural Science Fund grant #60496321

of this great intellectual edifice that is being built called Computational Complexity Theory. And if some young reader finds this subject so attractive as to devote himself in the study, we will feel amply rewarded.

With such a short survey, it is impossible to do justice to all the interesting and important topics in Complexity Theory. And in order to discuss some topics with a certain depth, it is necessary to choose only a small number of topics. We made this choice based on personal taste. This reflects the limitations of the authors, both in their insight as well as in their knowledge. We also apologize beforehand for any errors.

We assume the readers are acquainted with a basic knowledge of Theoretical Computer Science, such as the notion of computability (by Turing machines), time and space bounded computations and their respective hierarchy theorems, polynomial time reductions and NP-completeness theory. In short we assume the readers are familiar with the theory of computing at the level of knowledge represented by [21, 30].

The reader is likely to have also known some more recent developments such as a) the use of randomness in computation, b) randomized complexity classes, c) pseudorandomness as rigorously formulated by Blum-Micali [12] and Yao [73], as being indistinguishable from true randomness by any bounded computations such as polynomial time, and d) interactive proof systems.

2 The PCP Theorem

We speak of a computational task being "easy" if it can be accomplished by a Turing machine in polynomial time. This is the class P. The complexity class NP [19] was first defined as a proof system where a true statement can be easily certified to be so, while the system does not erroneously accept any false statement. E.g., consider the claim that a boolean formula φ is satisfiable. If φ is satisfiable, then this claim can be easily certified to be so by any satisfying assignment to φ. If φ is not satisfiable, then no purported satisfying assignment can convince us that φ is satisfiable.

Interactive Proof (IP) systems were proposed by Goldwasser, Micali and Rackoff in [27] as a probabilistic and interactive version of NP which is capable of proving an (apparently) wider class of statements. In such a system a Prover P is computationally unrestricted (called all powerful—logically, this means that his moves are existential) which interacts with a computationally restricted Verifier V (usually by probabilistic polynomial time). It is required that, when statement φ is true then P can convince V of its truth after a sequence of interactions. (This is called completeness requirement. In Logic, a proof system that can prove all true statements within the system is called complete.) Because V is polynomially bounded this sequence of interactions is, of necessity, also polynomially bounded. However, in this sequence of interactions the verifier V can also use random moves. For an IP system it is also required that when a statement is false, then any dishonest prover P' can not convince our verifier V that φ is true, with probability $> 1\%$, say. (This is called soundness requirement. In Logic, a proof system that only proves true statements within the system is called sound. Here 1% can be easily made exponentially small by a polynomial number of repetitions of the protocol.)

Here is an example of an IP system. It is for graph non-isomorphism (GNI). Clearly graph isomorphism (GI) is in NP: To prove two graphs are isomorphic, one only need to exhibit an isomorphism, which can be easily checked. However, it is not clear there is any universal property which proves two graphs are non-isomorphic and can be checked in polynomial time. Listing all possible maps and then verify that none is an isomorphism takes exponential time. Therefore GNI is not known to be in NP. However, consider the following game: A prover P claims that two labeled graphs A and B are non-isomorphic, and invites any verifier V to pick one of them in secret. Then V can apply a secret permutation to its vertices and present the resulting graph G back to P, and challenge P to identify which one of A and B was picked by V. V will accept P's claim of non-isomorphism iff P answers this challenge correctly. Note that if A and B are not isomorphic then the permuted graph G is isomorphic to exactly one of A and B, and P can succeed in this case. However, If A and B are isomorphic, and V chooses among A and B with equal probability and then applies a uniformly chosen permutation, then his graph G is a uniform sample from all graphs isomorphic to both A and B. Hence no cheating prover P' can meet the challenge with success probability greater than $1/2$. n parallel iterations (where they play this game independently in parallel n times, and V accepts iff all the answers P gives are correct) reduce the success probability of any cheating prover down to $1/2^n$.

It might appear that the use of secrecy in the random choice by V was essential. Independently, Babai [9] (see also [10]) proposed a public coin version of IP system called Arthur-Merlin games. Here the prover is called Merlin, and he can see all the coin flips the verifier (Arthur) makes. It was quite a surprise that Goldwasser and Sipser [28] showed that these two versions of IP systems are equivalent in computational power.

One interesting consequence of the fact that GNI having a constant round IP protocol is that GI is not NP-complete, unless the Polynomial Hierarchy PH collapses. Whether GI is NP-complete has been an open problem since the original paper on NP-completeness by Karp [33].

There followed a remarkable sequence of development in TCS starting in the late 80's and early 90's that eventually led to the celebrated PCP Theorem. Lund, Fortnow, Karloff and Nisan [42], following an initial breakthrough by Nisan, first showed that the permanent function Perm has an IP system. Perm is known to be hard for NP, in fact #P-hard, and thus hard for the entire Polynomial-time Hierarchy PH. Shortly afterwards A. Shamir [60] proved that the class of properties provable by IP is exactly the class PSPACE, the problems solvable in polynomial space.

The concept of Multiprover Interactive Proof (MIP) system [11] was introduced sometime after IP systems, and was apparently for purely intellectual purposes, with no practical applications in mind. After Shamir's result of IP = PSPACE, the power of MIP was also characterized as NEXP, non-deterministic exponential time. More importantly, the MIP concept eventually led to the concept of Probabilistically Checkable Proofs (PCP).

The next breakthrough came in a 1991 paper by Feige, Goldwasser, Lovász, Safra, and Szegedy [24]. In this paper, they first established a close relationship between the approximability of MAX-CLIQUE problem and two prover interactive proof systems. In very rough terms, the "proofs" the provers offer become vertices and their mutual consistencies become edges in a certain graph on which, either there is a large clique (a copy of the complete graph K_m) or there are only small cliques. This becomes a reduction, so that, in essence, one can reduce a decision problem to a gap problem (here the gap problem refers to the gap in the clique size, which is either small or large.) Thus any approximation algorithm with a performance guarantee within that gap translates into a decision algorithm for the original decision problem.

Many rounds of improvements eventually culminated in the following PCP Theorem.

Theorem 2.1 (Arora, Lund, Motwani, Sudan, Szegedy) *Fix any $\epsilon > 0$. Every set $S \in$ NP has a proof scheme as follows: There is a polynomial time proof checker C which can flip coins, such that*

- *$\forall x \in S$, there exists a "short proof" y, C will accept x with probability 1 after it examines a constant number of randomly chosen bits of y.*

- *$\forall x \notin S$, for any purported "short proof" y, C will reject x with probability $\geq 1 - \epsilon$ after it examines a constant number of randomly chosen bits of y.*

Building on this result, a great many inapproximability results followed, including MAXSAT, MAXCUT, Clique, Vertex Cover etc. E.g., it is shown that approximating Clique size to $n^{1-\epsilon}$ for any $\epsilon > 0$ is NP-hard. Research has since moved ahead with tremendous vigor. Today, the known results are quite amazingly broad and strong, and research is still advancing rapidly.

The interested reader can find much more information and many interesting pointers at the following web page
`http://www-cse.ucsd.edu/users/~mihir/pcp.html`

3 Reingold's Theorem

Space bounded computation is only second in importance to time bounded computations. The model of space bounded computation is an off-line Turing machine where the input tape is read only (size n) and does not count toward its space complexity. The space complexity is measured by the usage of the off-line read/write work tape. This way one can discuss sublinear $o(n)$ space complexity, in particular $O(\log n)$ or $(\log n)^{O(1)}$ bounded computations, which turn out to be extremely important and interesting.

Clearly both deterministic logspace L (= DSAPCE[$O(\log n)$]) and non-deterministic logspace NL (= NSAPCE[$O(\log n)$]) are subclasses of P. NL is represented, and in a technical sense in an essential way, by the following concrete problem called GAP (Graph Accessibility Problem): Given a directed graph G, and two vertices s and t, is there a directed path from s to t? Clearly one can solve GAP by Depth-First-Search, which takes linear time but it also takes linear space. Non-deterministically one can guess a path step by step, only remembering the current vertex visited. The question is can one do better deterministically in terms of space complexity?

Savitch [59] gave a recursive algorithm which solves GAP in space $O((\log n)^2)$. Unfortunately the time complexity is super-polynomial $n^{O(\log n)}$.

Major progress has been made for undirected graphs: Given an undirected graph G, is G connected? This problem is usually phrased as follows, and called USTCON: Given an undirected graph G and two vertices s and t, is there a path from s to t in G?

The first truly beautiful result on this problem is due to Aleliunas,, Karp, Lipton, Lovász and Rackoff [7] in 1979. They showed that a random walk of length $O(n^3)$ is most likely to visit all vertices of an undirected graph, no matter what n-vertex graph it is or which vertex the walk starts at. This gives a randomized algorithm as follows: To decide if one can reach from vertex s to t, one simply starts a random walk at s of length $O(n^3)$, and check that vertex t is ever reached. The remarkable fact is that this algorithm uses only $O(\log n)$ space—essentially it only needs to remember the name of the vertex it is currently visiting.

Subsequent improvements came with attempts to derandomize this algorithm, and more generally to derandomize all algorithms in randomized logspace RL or BPL. Here in analogy to randomized polynomial time classes, we define RL to consist of languages L acceptable by probabilistic logspace Turing Machines M such that whenever $x \in L$ then $\Pr[M(x) = 1] \geq 1/2$ and for $x \notin L$, $\Pr[M(x) = 1] = 0$. Here the probability Pr is taken over a sequence of polynomially many random bits, which the logspace Turing Machine M can access only on a one-way read-only tape. Machines accepting languages in RL have only one-sided error (when $x \in L$). BPL is the same except we allow two-sided error.

Nisan in 1992 [48] gave a pseudorandom generator which uses seed length $O((\log n)^2)$ and produces a polynomially long sequence of pseudorandom bits provably indistinguishable by any logspace bounded computation. While it did not improve the space bound of Savitch directly, Nisan showed that his generator can be used to prove an interesting result: The undirected graph

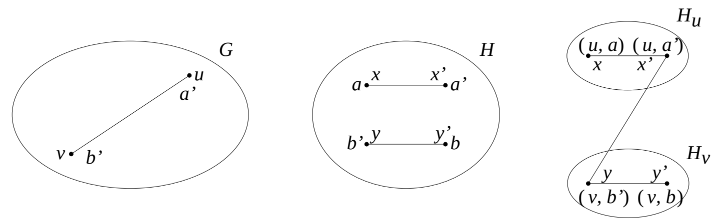

Figure 1: Zigzag graph product: (Left) An edge in G, (Middle) Two edges in H, (Right) A path of 3 edges in the cross product of G and H. It corresponds to the edge labelled by (x, y) from node (u, a) in $G \textcircled{z} H$

connectivity problem (in fact all of BPL) can be solved by an algorithm which runs in simultaneous polynomial time and poly-logarithmic space. This class of problems is called Steve's Class SC (after Steve Cook).

Then using Nisan's generator as a main ingredient, Nisan, Szemeredi and Wigderson [50] showed that USTCON can be solved in deterministic space complexity $O(\log^{1.5} n)$. This was followed by a sweeping result of Saks and Zhou [57] who showed that the entire randomized logspace BPL can be solved in deterministic space complexity $O(\log^{1.5} n)$. Subsequently Armoni, Ta-Shma, Wigderson and Zhou [8] showed that USTCON can be solved in space complexity $O(\log^{4/3} n)$. In terms of simultaneous time and space bound, the best simulation result for the class of randomized logspace RL or BPL is due to Cai, Chakaravarthy and Melkebeek [14]. They show that, for all $0 \le \alpha \le 0.5$, $\mathsf{BPL} \subseteq \mathsf{DTISP}[n^{O(\log^{0.5-\alpha} n)}, O(\log^{1.5+\alpha} n)]$, where $\mathsf{DTISP}[t, s]$ denotes the complexity class of problems decidable in simultaneous time t and space s. With $\alpha = 0$, we get the Saks-Zhou result, and with $\alpha = 0.5$, we get Nisan's SC result; thus it gives a smooth transition between the bounds of Nisan and Saks-Zhou.

The most recent breakthrough is a brilliant result of O. Reingold, who gave an algorithm for USTCON with optimal space complexity $O(\log n)$.

Reingold's algorithm uses a graph product operation called the zig-zag product, which had been used to construct expander graphs. In Reingold's algorithm, a preliminary step transforms G into a D-regular non-bipartite graph for some well chosen constant D. This step is easily carried out in Logspace. Instead of the usual method of specifying a graph by its adjacency matrix, we will further label every edge $e = \{u, v\}$ incident to a vertex u, at the end of u, by an integer between 1 and $D = \deg u$. For the input graph G, this ordering of edges incident to u can be easily computed in Logspace, by assigning edges numbers from 1 to D according to the numerical ordering of the vertex v. We formalize this information by saying our graph G is specified by a *rotation map* $Rot_G : [n] \times [D] \to [n] \times [D]$ such that $Rot_G(u, a) = (v, b)$ if the a-th edge incident to u is $e = \{u, v\}$ which leads to v and this edge is the b-th edge incident to v.

Given an n-node, D-regular graph G and a D-node, d-regular graph H, the zig-zag product $G \textcircled{z} H$ was introduced in [56] as a method to construct expanders. The graph $G \textcircled{z} H$ has its vertex set $[n] \times [D]$, and every vertex has d^2 edges labelled by $(x, y) \in [d] \times [d]$. It is defined by the following *rotation map* $Rot_{G \textcircled{z} H}$:

$$Rot_{G \textcircled{z} H}((u, a), (x, y)) = ((v, b'), (y', x')),$$

where $Rot_H(a, x) = (a', x')$, $Rot_G(u, a') = (v, b')$, and $Rot_H(b', y) = (b, y')$, see Figure 1 for an illustration. Note that in reverse, $Rot_{G \textcircled{z} H}((v, b'), (y', x')) = ((u, a), (x, y))$.

In [56] a remarkable property concerning the spectral gap (i.e., the difference between 1 and its second largest eigenvalue of the normalized adjacency matrix) is proved which has the following direct consequence [55]:

$$1 - \lambda(G ⓩ H) \geq \frac{1}{2}(1 - \lambda(H)^2)(1 - \lambda(G))$$

where $\lambda(X)$ denotes the second largest eigenvalue of the normalized adjacency matrix of graph X. Essentially this shows that if H is chosen to be a good expander (i.e., a graph with a small second largest eigenvalue), then $G ⓩ H$ will have a spectral gap not much smaller than that of G. Meanwhile a powering of X will increase the spectral gap of X, since $\lambda(X^k) = (\lambda(X))^k$. Reingold chose $D = d^{16}$ and an appropriate H with D vertices. Then $(G ⓩ H)^8$ is D-regular again, and has an increased spectral gap than G by the spectral gap formula above.

The main part of Reingold's algorithm is to repeatedly apply the zig-zag product and graph powering to the input graph G, to obtain a G' after $O(\log n)$ steps. G' is connected iff G is connected. Now each connected component of G' is an expander with second largest eigenvalue at most 1/2. Being an expander, this implies that any pair of nodes in the same connected component of G' are joined by a path of length at most $O(\log n)$. Thus, on G', one can solve connectivity problem in deterministic logspace.

The zig-zag product operation has turned out to be extremely powerful. It was initially used for the construction of explicit families of expander graphs, which is a rich subject with enormously important applications by itself. But we will not go into that in this article. There has already been some important follow ups. Dinur [23] has shown how to use a similar approach to give a simpler proof of the PCP Theorem, which has some stronger parameters.

There is at least one other important result regarding space complexity in the last twenty years which we should mention. Immerman and Szelepcsényi [31] [63] independently proved that NL is closed under complement. Thus NL = coNL. If NL is the space analogue of NP for space complexity, then the alternating space hierarchy, the space analogue of PH, collapses to NL. The analogous collapse for non-deterministic polynomial time would be NP = coNP, the opposite of which is conjectured to be true by most researchers.

There is a subfield of Complexity Theory, called Proof Complexity, which is devoted to the goal of proving NP $\neq$ coNP. Currently Proof Complexity mainly produces strong lower bounds for very weak proof systems. Even so, these proof techniques have been rather powerful. For example, it is well established that for Resolution, a proof system widely used in AI, there are exponential lower bounds for very simple formulas. These exponential lower bounds hold even for much more powerful proof systems than Resolution. Therefore, even in more practical areas of work, it is unwise to pin too much hope on proof systems such as Resolution.

4 AKS Primality Testing

Perhaps the problem of Primality Testing and the related problem of Factoring are the single most comprehensible computational problem, if a complexity theorist has to explain to someone, who is generally educated but has no particular knowledge of this subject, "What is complexity theory".

The problem of Primality Testing is the following: Given an integer N, decide whether N is a prime. The question is whether there is an efficient algorithm that will always decide this question correctly for any input integer N. (Of course, the complexity of any algorithm is measured in

$\lceil \log_2 N \rceil$, the binary length of the input N.) The problem of Factoring is to find the unique prime factorization of an input N. Clearly if one can factor N, one can tell if it is a prime.

This problem is clearly computable, as the Sieve of Eratosthenes 200 BC already shows how to factor N by testing divisibility of all integers up to $\sqrt{N}$. However, this procedure takes exponential time (in input length $\lceil \log_2 N \rceil$.)

The search for an efficient algorithm was already discussed by Gauss in his *Disquisitiones Arithmeticae* in 1801. Clearly if a number N is not a prime, there is an easy proof: just give a non-trivial factorization. Thus Primality is in coNP. It is also known that Primality is in NP, a result due to Pratt [52]. In the mid 1970s, a pair of algorithms, one due to Solovay and Strassen [62], and one due to Rabin [53] based on an earlier algorithm of Gary Miller [46], showed that there are probabilistic algorithms for Primality, which works for every input with high probability. More precisely, their probabilistic algorithms run in polynomial time, and guarantees the following: If N is composite, then with probability at least 99% the algorithm will say so; if N is prime, then the algorithm will always answer "looks like a prime". (There is a small error probability, $\leq 1\%$, when N is composite, yet the algorithm says "looks like a prime". As usual, the constant $\leq 1\%$ can be easily reduced to exponentially small by running the algorithm a polynomial number of times.) In complexity terms, this means Primality $\in$ coRP. The running time of these probabilistic algorithms can be shown to be $O(n^2 \log n \log\log n)$. (We will write $\tilde{O}(n^2)$ to indicate this, ignoring poly-log factors.)

A great theoretical achievement was accomplished in early 1990s when Adleman and Huang [1] showed that there is a polynomial time probabilistic algorithm for Primality which never makes any definite errors. More precisely, upon any input N, it will only answer, in polynomial time, either "prime", or "composite", or "unclear", and, for any N, with probability at least 99% (or with an exponentially small error probability by repetition) it does not answer "unclear". Such an algorithm is called Las-Vegas, and in complexity terms, this means Primality $\in$ ZPP, Zero-error Probabilistic Polynomial time. The Adleman-Huang algorithm uses some most advanced mathematics such as Jacobian varieties, using work by the Fields Medalist G. Faltings. But their algorithm is of purely theoretical interest, since it has an estimated running time of $O(n^{30})$.

On August 6, 2002, Manindra Agrawal, Neeraj Kayal and Nitin Saxena (AKS) released on the Internet a manuscript which proves that Primality is in P. This is a sensational achievement. Even the New York Times covered this just two days later. (US newspapers seldom cover anything that is a true mathematical achievement.)

The status of the deterministic complexity of Primality prior to the AKS work is as follows. Using an unproven yet widely accepted number theoretic hypothesis called the Extended Riemann Hythothesis (ERH), one can show that the randomized Primality test can be done deterministically in polynomial time. The ERH implies a great deal of "regularity" on various number theoretic distributions, and Miller's original Primality test in fact was deterministic, whose polynomial time complexity is only proved using ERH. (It was Rabin's insight to substitute this reliance on unproven hypothesis by randomization.) A year earlier than AKS, Agrawal and colleagues had searched for alternative randomized algorithms for Primality. The idea was to find some alternative algorithms which can be derandomized, without using any unproven number theoretic assumptions. It is a remarkable fact (and a bit of irony) that while the Solovay-Strassen and Miller-Rabin randomized algorithms for Primality ushered in an era of tremendous research activity on randomized algorithm in general, the problem Primality itself is actually shown to be in P.

The AKS proof uses a number theoretic estimate proven by Fouvry in 1980 (without any unproven hypothesis such as the Extended Riemann Hythothesis (ERH).) There was some non-constructiveness in that Fouvry estimate. While Fouvry's proof is unconditional, it asserts some number theoretic property which is true for all sufficiently large N. Exactly how large must N be, the Fouvry estimate does not say, but the AKS algorithm uses this bound. If the ERH is true, then this bound can be effectively computed (but in that case there are already better deterministic algorithms for Primality.) If the ERH is false, then this bound can also be effectively estimated, but in this case it seems to depend on the smallest counter example to ERH, which is, of course, unknown.

In their journal paper in Ann. of Math. in 2004 [2], AKS managed to circumvent this problem. using an idea of Lenstra, they found a constructive algorithm without using the Fouvry result which runs in $\tilde{O}(n^{10.5})$, and also to reduce the running time of the Fouvry-based algorithm to $\tilde{O}(n^{7.5})$. This has been improved by Lenstra and Pomerance (2005) to $\tilde{O}(n^6)$ for the constructive version of the algorithm. Also Bernstein has developed a ZPP algorithm for Factoring in time $O(n^{4+\epsilon})$, which is a major improvement over the Adleman-Huang algorithm.

We will not go any further into the technical details of these algorithms, because much has been written on this. The readers are encouraged to consult the original papers, as well as a write up by Aaronson

http://www.scottaaronson.com/writings/prime.pdf

See also http://mathworld.wolfram.com/AKSPrimalityTest.html One can find there many citations and pointers to the articles in downloadable form. In particular, one can consult an article by Granville [22], and a forthcoming 2nd Edition of a book by Crandall and Pomerance.

Those are the good news for Primality. Perhaps the more interesting problem is Factoring. Note that in all these algorithms, when they declare a certain number composite, it does not provide its prime factorizations. The best general Factoring algorithm runs in time $O(c^{n^{1/3}(\log n)^{2/3}})$, where $n = \log N$, and $c \approx 1.902$. (The proof of the running time is not totally rigorous mathematically.) Shor [61] gave a polynomial expected time quantum algorithm for Factoring. Thus a quantum computer, if one can be built, can break the RSA scheme. This suggests that Factoring is not NP-hard. There is another indication that the problem is not NP-hard, due to its membership in the complexity class UP, but we will not discuss that any further. The complexity of Factoring is still wide open. There are no completeness result or meaningful lower bounds.

5 Lattice Problems

Complexity Theory primarily focuses on worst-case complexity of computational problems. When we say, it is conjectured that Factoring is not solvable in P, we mean that there is no algorithm that will factor every integer in polynomial time. However, in many situations, especially in cryptography, the worst-case complexity measure is not the right measure. E.g., the security of RSA public key system using a composite number $N = pq$ as its modulus is based on the intractability of factoring N, where N is the product of two somewhat randomly chosen primes p and q. Here one would like to have some evidence that factoring is hard on the average for $N = pq$. This average-case hardness clearly implies the hardness of Factoring in the worst-case (since any universal factoring algorithm which can factor any N certainly can factor numbers of the form $N = pq$ on average.) As yet, there appears to have no hope to prove the hardness of Factoring either in the worst-case or in the average-case. However, one might hope to prove a logical implication, that if the problem is hard in the worst-case (which we strongly believe) then it is also hard in the average-case (which we need in cryptography). Such a proof, however, is still unavailable for Factoring.

So can one prove any such conditional hardness for a problem in NP? In recent work by Miklós Ajtai, such a reduction is established for the first time for certain lattice problems, which are in NP.

A lattice is a discrete additive subgroup in some $\mathbf{R}^n$. Discreteness means that every lattice point is an isolated point in the topology of $\mathbf{R}^n$. An alternative definition is that a lattice consists of all the integral linear combinations of a set of linearly independent vectors,

$$L = \{\sum_i n_i b_i \mid n_i \in \mathbf{Z}, \text{ for all } i\},$$

where the vectors b_i's are linearly independent over $\mathbf{R}$. Such a set of generating vectors are called a basis. The dimension of the linear span, or equivalently the number of b_i's in a basis is the rank (or dimension) of the lattice, and is denoted by $\dim L$.

The Shortest Vector Problem (SVP) is the following: Given a lattice by its basis vectors, find the shortest non-zero vector in the lattice. There are several other related problems. The Closest Vector Problem (CVP) is as follows: Given a lattice by its basis vectors and a point x in space, find the closest lattice vector to x. One also studies the shortest basis problem which asks for a basis which is shortest in some well defined sense.

These lattice problems have many important applications. This includes the polynomial time algorithm to factor polynomials, breaking certain public-key cryptosystems, solving integer programming in fixed dimensions in polynomial time, and solving simultaneous Diophantine approximations. Research in the algorithmic aspects of lattice problems has been active in the past, especially following Lenstra-Lenstra-Lovász basis reduction algorithm in 1982 [38, 39, 40]. The recent wave of activity and interest can be traced in large part to two seminal papers written by Ajtai in 1996 and in 1997 respectively.

In his 1996 paper [4], Ajtai found a remarkable worst-case to average-case reduction for some versions of the shortest lattice vector problem (SVP). Such a connection is not known to hold for any other problem in NP believed to be outside P. In his 1997 paper [5], building on previous work by Adleman, Ajtai further proved the NP-hardness of SVP (in L_2 norm), under randomized reduction. The NP-hardness of SVP in L_2 norm has been a long standing open problem. Stimulated by these breakthroughs, many researchers have obtained new and interesting results for these and other lattice problems [6, 17, 25, 26, 44, 47].

The worst-case to average-case reduction for these lattice problems by Ajtai give the following reduction: Suppose there is a polynomial time algorithm $\mathcal{A}$ that can solve approximately a random instance of a certain version of SVP, with probability at least $1/n^c$ over the random choices of the lattice, then there is a probabilistic algorithm $\mathcal{A}$ which can solve approximately some other versions SVP (including the shortest basis problem) with probability $1 - 2^{-n}$ for every lattice.

More technically, we need to define a family of lattices as follows: Let n, m and q be arbitrary integers. Let $\mathbf{Z}_q^{n\times m}$ denote the set of $n \times m$ matrices over $\mathbf{Z}_q$, and let $\Omega_{n,m,q}$ denote the uniform distribution on $\mathbf{Z}_q^{n\times m}$. For any $X \in \mathbf{Z}_q^{n\times m}$, the set $\Lambda(X) = \{y \in \mathbf{Z}^m \mid Xy \equiv 0 \bmod q\}$ (where the congruence is component-wise) defines a lattice of dimension m. Let $\Lambda = \Lambda_{n,m,q}$ denote the probability space of lattices consisting of $\Lambda(X)$ by choosing X according to $\Omega_{n,m,q}$.

By Minkowski's First Theorem, it can be shown that

$$\forall c \;\exists c' \text{ s.t. } \forall \Lambda(X) \in \Lambda_{n,c'n,n^c} \;\exists v \;(v \in \Lambda(X) \text{ and } 0 < ||v|| \leq n).$$

In the following theorem, we will use the standard notation for Minkowski's successive minima, namely $\lambda_1(L)$ is the length of the shortest non-zero vector v_1 in L, $\lambda_2(L)$ is the length of the shortest vector in L not in the linear span of v_1, etc.

Now we can state Ajtai's theorem.

Theorem 5.1 (Ajtai) *Suppose there is a probabilistic polynomial time algorithm $\mathcal{A}$ such that for all n, when given a random lattice $\Lambda(X) \in \Lambda_{n,m,q}$ where $m = \alpha n \log n$ and $q = n^\beta$ for appropriate* *constants α, β, returns with probability $\frac{1}{n^{O(1)}}$, a vector of $\Lambda(X)$ of length $\leq n$, then there exists a probabilistic polynomial time algorithm $\mathcal{B}$ such that for all N, when given a basis for an arbitrary lattice L of dimension N, performs the following with probability $\geq 1 - 2^{-N}$:*

1. *Finds a basis $b_1, \ldots, b_N$ for L such that $\max_{i=1}^{N} ||b_i||$ is up to a factor N^{c_1} the best possible for L.*
2. *Finds an estimate $\tilde{\lambda}$ of $\lambda_1(L)$ (the shortest vector length in L) such that,*
$$\frac{\lambda_1(L)}{N^{c_2}} \leq \tilde{\lambda} \leq \lambda_1(L),$$
3. *Finds the unique shortest vector $\pm v$ of L, if L has an N^{c_3}-unique shortest vector, i.e. $\lambda_2(L) \geq N^{c_3} \cdot \lambda_1(L)$,*

where c_1, c_2, c_3 are absolute constants.

These constants c_1, c_2, c_3 are called connection factors. They control the tightness of this worst-case to average-case reduction. In particular c_1 is the basic connection factor.

Much effort has been given to the improvement of this connection factor. The current best connection factor is due to Micciancio and Regev [45].

In the other seminal paper by Ajtai in 1997, he showed that SVP is NP-hard, under L_2-norm, as it has long been conjectured. His reduction is randomized. Thus this problem is not solvable in randomized polynomial time if RP $\neq$ NP.

Even certain approximation to the SVP problem is also shown to be NP-hard under randomized reduction. Ajtai's original paper gave an inapproximability factor of $1+1/2^{n^k}$, a rather weak bound. This has been improved in successive papers. Currently the best bound is due to Khot [34]: for any constant $c > 0$, approximation to the SVP problem within a factor c is NP-hard under randomized reduction, thus not solvable in P time unless RP $\neq$ NP.

There is a further twist on this saga. Goldreich and Goldwasser [25] have shown that approximating to a factor of $O(\sqrt{n/\log n})$ to both SVP and CVP are not NP-hard, unless PH collapses to its second level Σ_2^p. Aharonov and Regev [3] have shown that approximating to a factor of $O(\sqrt{n})$ is not NP-hard, unless PH collapses to its first level NP = coNP.

There is also an interesting twist on the unique shortest vector problem, introduced in Ajtai's paper. This problem has been the basis of the public-key cryptosystem designed by Ajtai and Dwork [6], which is the first public-key cryptosystem with a provable guarantee on its security assuming only a worst-case complexity assumption. For this problem Cai [18] showed that $n^{1/4}$-unique shortest vector problem is not NP-hard, unless PH collapses to Σ_2^p. In [13] an analogue of Ajtai's theorem for CVP is proved.

Once upon a time, life was simple (or so it appeared to be): To prove a certain problem is easy, one simply gives a good algorithm and shows that it runs in polynomial time. To show certain problems are hard, one shows they are NP-complete or NP-hard. In a foundational sense, being NP-hard does not prove that it is not solvable in P; it only proves that it is unlikely to be easy, provided that P $\neq$ NP, an extreme complexity theoretic collapse. We now have methods to prove certain problems are not NP-hard, unless some unlikely event occurs, namely the Polynomial Hierarchy PH collapses. Thus, in a foundational sense, such proof shows that "it is unlikely that the problem is unlikely to be easy by way of NP-hardness." Note that this is not quite the same as it being easy.

6 Pseudorandomness and Extractors

Monte-Carlo algorithms have been used for many years since the time of Nicholas Metropolis and Stanislaw Ulam [43]. Closely related is the notion of a pseudorandom generator. In the traditional treatment, e.g. in the multi volume treatise by Knuth [37], the study of pseudorandom generators as used in algorithms was given substantial coverage, mainly in terms of various statistical tests. Here the set up is as follows. One has some particular process to generate a sequence of numbers or bits, such as a linear congruential generator. Then we can design a certain statistical test, and say that a sequence of numbers generated by this process passes the statistical test, if it behaves similarly to the statistical test as a true random sequence. If this is so, the generating process is called a pseudorandom generator. As an example of a simple minded statistical test, one can count the number of bits 0's and 1's, and a pseudorandom sequence should have roughly the same number of 0's and 1's.

This ad hoc view of pseudorandom generator was revolutionized by the work of Blum and Micali [12] and by Yao [73] in 1982. Assume a complexity theoretic hardness on the problem of Discrete Logarithm (This is the following problem: given a prime modulus p and a generator $g \in \mathbf{Z}_p^*$, for $y = g^x \bmod p$ find the index x), Blum and Micali gave a pseudorandom generator which provably has the following property to any polynomial time bounded computer: Given any initial segment of the bits of the sequence, the next bit is essentially unpredictable. It was Yao who gave what has become the standard definition of pseudorandomness: No polynomial time bounded computer M can tell apart the pseudorandom sequence and true random sequence. More precisely, suppose G_n is a generator mapping $\{0,1\}^{k(n)} \to \{0,1\}^n$, $n \in \mathbf{N}$, then for all polynomial $p(n)$ and for all polynomial time bounded TM M, for all sufficiently large n,

$$\left|\Pr_{x\in\{0,1\}^n}[M(x)=1] - \Pr_{y\in\{0,1\}^{k(n)}}[M(G_n(y))=1]\right| < \frac{1}{p(n)}.$$

Yao proved that his general notion of pseudorandomness is equivalent to Blum-Micali's next-bit unpredictability against all polynomial time bounded TM. He also showed that starting with any one-way permutation one can construct a polynomial time computable pseudorandom generator. A function is called a one-way function if it is easy to compute but difficult to invert, all formally defined in terms of polynomial time computability. The Discrete Logarithm function used by Blum and Micali is an example of a (presumed) one-way permutation. This idea of passing all polynomial time statistical tests, and its equivalence to the next-bit unpredictability test, has been the foundation of all the development that follows.

After a great deal of successive improvements by many researchers, finally Håstad, Impagliazzo, Levin and Luby [29] succeeded in proving the following definitive theorem: Starting with any one-way function, one can construct a pseudorandom generator.

This proof has two main ideas, besides many technical details. The first is the idea of Goldreich and Levin of a hardcore bit. They showed that if f is a one-way function mapping $\{0,1\}^n$ to $\{0,1\}^n$, then a random inner product bit $X \cdot Y$ is unpredictable (the hardcore bit) to any polynomial time adversary, given a uniformly chosen random $Y \in \{0,1\}^n$, and $f(X) \in \{0,1\}^n$ where X is chosen uniformly random (but hidden to the adversary) from $\{0,1\}^n$. The second main idea in this elaborate proof is the idea of universal hashing (which is due to Carter and Wegman from 1979 [20], an idea originated within Theoretical Computer Science and found many practical applications all over computer science.)

Another important theorem that came out of Yao's ground breaking work on the pseudorandomness is his famous XOR lemma. This essentially says that if $f : \{0,1\}^n \to \{0,1\}$ and $f(X)$ is slightly unpredictable to a polynomial time adversary, for uniformly chosen $X \in \{0,1\}^n$, then the Exclusive-Or bit $f(X_1) \oplus f(X_2) \oplus \ldots \oplus f(X_m)$ is exponentially (in m) more unpredictable.

We now go on to discuss extractor constructions. This is a topic that has seen some of the most brilliant ideas and technical depth, and is still very active. We will only be able to describe a small part of it.

In the early 1980s, people started to be interested in the question of what happens if the random source given to a randomized algorithm is not really random, but correlated in some way. For example, maybe the sequence is generated by a linear congruential generator. (In most software systems whenever you call a RANDOM number in a program, this linear congruential generator or some versions of it is most likely what you get.) Or maybe the sequence is generated by an underlying random process, such as a Markov chain, which has a bias depending on the underlying (hidden) state. Or maybe the sequence is generated by a physical process which has a certain guarantee (based on quantum mechanics?) that every next bit X_i generated has a certain amount of randomness, such as $\Pr[X_i = 1] \in [\epsilon, 1-\epsilon]$, but nevertheless is dependent on the previous bits (or even controlled by an adversary).

There are results in both the positive and negative directions. As an example of an early result in the positive direction, Santha and U. Vazirani [58], and U. Vazirani and V. Vazirani [72] showed how to construct guaranteed pseudorandom sequences from "slightly-random" sources. (We omit their formal definitions here as they are generally subsumed by the subsequent extractor machineries.) On the negative side, it was shown that linear congruential generators are insufficient for Quicksort.

It was Nisan and Zuckerman in 1996 [51] who first realized the unifying concept and defined the notion of an Extractor. We first define min-entropy: Given a distribution X on $\{0,1\}^n$, the min-entropy, $H_\infty(X)$, is defined as $H_\infty(X) = \min_x(-\log_2(\Pr[X = x]))$. Thus, $2^{-H_\infty(X)}$ is the largest probability assigned to any point. Intuitively $H_\infty(X)$ represents the number of true random bits contained in X.

An extractor extracts the randomness contained in a random source X, with the help of some additional true random bits.

Definition 6.1 *An (n,d,k,m,ϵ)-extractor is a function* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$*, such that whenever a distribution X on $\{0,1\}^n$ has $H_\infty(X) \geq k$,*

$$||\mathrm{Ext}(X, U_d) - U_m|| \leq \epsilon.$$

Here U_ℓ is the uniform distribution on $\{0,1\}^\ell$, and $||\cdot||$ denotes statistical distance. The statistical distance between two distributions X and Y on the same set is defined to be $||X - Y|| = \max_D |\Pr_X[D] - \Pr_Y[D]|$. Ideally, we would like to maximize m to be close to k, while minimize d and ϵ.

It can be shown that without using additional random bits U_d, this is impossible. Nonconstructively it can be shown that for every $n, k \leq n$ and $\epsilon > 0$, there exists a (n,d,k,m,ϵ)-extractor with $m = k$ and $d = O(\log(n/\epsilon))$. However the challenging task is to construct explicit extractors, computable in polynomial time.

Explicit constructions of extractors have many applications. Among them: simulation of randomized algorithms, constructions of oblivious samplers, randomness-efficient error reductions, explicit constructions of expanders, sorting networks, hardness of approximation, and pseudorandom generators for space-bounded computations.

In 2003, Lu, Reingold, Vadhan, and Wigderson [41] gave a construction that is optimal up to some constant factors: for every constants $\alpha > 0, \epsilon > 0$, they gave a (n,d,k,m,ϵ)-extractor, for all

$n, k \le n$ and, $m = (1-\alpha)k$ and $d = O(\log n)$. That is, their extractor can extract $m = (1-\alpha)k$ bits of entropy from any source with $H_\infty(X) \ge k$, using only $O(\log n)$ bits of additional random bits.

However, their construction uses many previous constructions and techniques, and these techniques have undergone many rounds of improvements, that it is infeasible to give a coherent account in a few pages. Instead we will give an account of an elegant construction by Trevisan [65], as improved by Raz, Reingold and Vadhan [54]. This construction introduced techniques that are important in many aspects, and it is representative of the more recent research in Computational Complexity Theory, such as the use of finite fields, the connection to coding theory, and the use of combinatorial designs.

There are two main ingredients of this construction. The first is the use of error correcting code. The second is a combinatorial design, first used by Nisan and Wigderson.

We first introduce a lemma. Consider a set $S \subseteq \{0,1\}^n$ of codewords for some error-correcting code with relative distance d/n. We would like to bound the number of codewords in a ball of radius w. For $x, y \in \{0,1\}^n$, let $d(x,y)$ denote the Hamming distance between x and y, and let $wt(x) = d(x, 0^n)$ denote the Hamming weight of x.

Lemma 6.1 *Let $S \subseteq \{0,1\}^n$. Assume for all $x, y \in S$, $x \ne y$, we have $d(x,y) \ge d$. Let $0 \le w \le n$, and let $N = |\{x \in S : wt(x) \le w\}|$. Then the following upper bound holds for N:*

$$N \le \frac{1-2\beta}{4\alpha^2 - 2\beta},$$

where $\alpha = 1/2 - w/n$, and $\beta = 1/2 - d/n$, and we assume $\alpha^2 > \beta/2$.

Note that this lemma in fact applies to any ball of radius w: We can simply shift the set S and the ball so that the ball is centered at the origin 0^n.

To prove this, consider the following sum

$$\Sigma = \sum_{i,j=1}^{N} \langle v_i, v_j \rangle$$

where $v_1, \ldots, v_N$ are the points in $\{x \in S : wt(x) \le w\}$, and where the inner product $\langle v_i, v_j \rangle = \sum_{k=1}^{n} v_{ik} v_{jk}$. Let W_i denote the set of 1-bits in v_i, and $w_i = |W_i| = wt(v_i)$. Notice that $\langle v_i, v_j \rangle = |W_i \cap W_j|$ and also that $d(v_i, v_j) = w_i + w_j - 2|W_i \cap W_j|$ we derive $\langle v_i, v_j \rangle = \frac{1}{2}(w_i + w_j - d(v_i, v_j))$. Furthermore, because $d(v_i, v_j) \ge d$ we have $\langle v_i, v_j \rangle \le \frac{1}{2}(wt(v_i) + wt(v_j) - d)$ for all $i \ne j$. Therefore we can rewrite Σ, letting $\bar{w}$ denote the average weight over all v_i:

$$\begin{aligned}
\Sigma &= \sum_{i=1}^{N} \langle v_i, v_i \rangle + \sum_{1 \le i \ne j \le N} \langle v_i, v_j \rangle \\
&\le \sum_{i=1}^{N} w_i + \sum_{1 \le i \ne j \le N} \frac{wt(v_i) + wt(v_j) - d}{2} \\
&= N\bar{w} + N(N-1)(\bar{w} - \frac{d}{2}).
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\Sigma &= \sum_{k=1}^{n} (\sum_{i=1}^{N} v_{ik})(\sum_{j=1}^{N} v_{jk}) \\
&= \sum_{k=1}^{n} (\sum_{i=1}^{N} v_{ik})^2.
\end{aligned}$$

Let $X_k = \sum_{i=1}^{N} v_{ik}$ be the number of v_i that have a 1 in the k^{th} position. Then $\sum_k X_k$ is the total number of 1's over all v_i, and therefore $\sum_k X_k = N\bar{w}$. By convexity, we obtain the lower bound:

$$\Sigma = \sum_{k=1}^{n} X_k^2 \geq n \left(\frac{\sum_k X_k}{n} \right)^2 = \frac{(N\bar{w})^2}{n}.$$

Combining these two bounds, we get

$$N \leq \frac{\frac{d}{2n}}{(\frac{\bar{w}}{n})^2 - (\frac{\bar{w}}{n}) + \frac{d}{2n}} = \frac{1-2\beta}{4\alpha'^2 - 2\beta},$$

where $\beta = \frac{1}{2} - \frac{d}{n}$ and $\alpha' = \frac{1}{2} - \frac{\bar{w}}{n}$. As each $w_i \leq w$, we have $\bar{w} \leq w$, and so for $\alpha = \frac{1}{2} - \frac{w}{n}$ we have $\alpha' \geq \alpha$, and it follows that

$$N \leq \frac{1-2\beta}{4\alpha^2 - 2\beta}.$$

Setting $\alpha = \sqrt{\gamma}$ and $\beta = \gamma/2$. We get:

Corollary 6.1 *If every pair of points in $S \subseteq \{0,1\}^n$ has relative distance at least $\frac{1-\gamma}{2}$, then every ball of relative radius at most $1/2 - \sqrt{\gamma}$ contains at most $1/3\gamma < 1/\gamma$ points in S.*

We briefly review the Reed-Soloman Code. Let $\mathbf{F} = GF[2^m]$, and let $\alpha_0, \alpha_1, \ldots, \alpha_{k-1} \in \mathbf{F}$. The Reed-Solomon (RS) encoding of $\alpha = \alpha_0 \ldots \alpha_{k-1}$ interprets the α_i as coefficients of a polynomial of degree $k-1$; that is, α defines the polynomial

$$f_\alpha(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \ldots + \alpha_{k-1} x^{k-1}.$$

Then the RS encoding of α, RS(α), is the concatenation of function values of f_α evaluated at all points a_i in the field $\mathbf{F}$:

$$\mathrm{RS} : \alpha \mapsto (f_\alpha(a_1), f_\alpha(a_2), \ldots, f_\alpha(a_{|\mathbf{F}|})).$$

Equivalently, we may consider the $2^m \times k$ matrix

$$M = \begin{pmatrix} 1 & a_1 & a_1^2 & a_1^3 & \ldots & a_1^{k-1} \\ 1 & a_2 & a_2^2 & a_2^3 & \ldots & a_2^{k-1} \\ 1 & a_3 & a_3^2 & a_3^3 & \ldots & a_3^{k-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}$$

and let $\bar{\alpha} = (\alpha_0, \ldots, \alpha_{k-1})^T$, so that the RS encoding of α is $M\bar{\alpha}$. Note that if $\alpha \neq \beta$ then the corresponding polynomials $f_\alpha \neq f_\beta$ can agree on at most $k-1$ points. It follows that the RS encodings RS(α) and RS(β) must disagree on at least $|\mathbf{F}| - (k-1) = 2^m - (k-1)$ entries.

Our goal is to design a binary code of polynomial length (and polynomial time computable) with the property that the relative distance of code words is almost $1/2$. Even though the RS code has the property that two distinct words differ on most entries, measured as a binary code, two different values in $\mathbf{F}$ may still have many identical bit positions. One easy method to get around this is via a process called code concatenation.

Thus we introduce the Hadamard code (Had). This code has 2^m code words each of length 2^m. Consider a $2^m \times 2^m$ matrix H_m with 0, 1 entries. Each row and column of $H_m = (h_{x,y})$ is indexed by $x, y \in \mathbf{Z}_2^m$, and has the value $h_{x,y} = \langle x, y \rangle \bmod 2$, the inner product mod 2. Each m bit string $x \in \mathbf{Z}_2^m$ is coded as a 2^m long code word $\mathrm{Had}(x) = (h_{x,y})_{y \in \mathbf{Z}_2^m}$. Note that whenever $x \neq x'$, Had(x) and Had(x') differ by exactly half of 2^m entries.

Now we can consider the concatenation of the Reed-Soloman code with the Hadamard code. Given $\alpha \in \{0,1\}^n$, the Reed-Solomon code treats α as specifying the coefficients of a degree $d \doteq \lceil n/m \rceil - 1$ polynomial over a field $\mathbf{F}$ where $|\mathbf{F}| = 2^m$. For $\alpha \neq \alpha'$, the code words $\mathrm{RS}(\alpha)$ and $\mathrm{RS}(\alpha')$ differ on at least $2^m - d$ entries. By taking

$$\mathrm{EC}_{n,m}(\alpha) = (\mathrm{Had}(p_\alpha(a_1)), \mathrm{Had}(p_\alpha(a_2)), \ldots, \mathrm{Had}(p_\alpha(a_{|\mathbf{F}|}))),$$

we get a binary code of length 2^{2m} and relative distance at least $\frac{(2^m-d)2^m/2}{2^{2m}} \geq \frac{1}{2} - \frac{n}{2^{m+1}}$. For any $\delta > 0$, we let $m = \lceil \log(n/\delta^2) \rceil$, then $\overline{n} = 2^{2m} = O(n^2/\delta^4)$ is the length of the code, and the relative distance $\geq \frac{1}{2} - \frac{\delta^2}{2}$. We will denote this code $\mathrm{EC}_{n,m}$ as $\mathrm{EC}_{n,\delta}(\alpha)$ from now. Therefore, by the Corollary, any ball of relative radius $\frac{1}{2} - \delta$ contains at most $\frac{1}{\delta^2}$ points from the code.

Lemma 6.2 *For all $n \in \mathbf{N}$ and $\delta > 0$, there exists an error-correcting code $\mathrm{EC}_{n,\delta} : \{0,1\}^n \to \{0,1\}^{\bar{n}}$, where $\bar{n} = poly(n, \frac{1}{\delta})$, such that every Hamming ball of relative radius $\frac{1}{2} - \delta$ in $\{0,1\}^n$ contains at most $\frac{1}{\delta^2}$ codewords. Furthermore, $\mathrm{EC}_{n,\delta}$ can be computed in time $poly(n, \frac{1}{\delta})$, and $\bar{n}$ is a power of 2.*

A second ingredient of this extractor construction is the idea of a combinatorial design. Trevisan's construction directly uses a design first used by Nisan and Widgerson. The RRV construction (by Raz, Reingold and Vadhan) uses a slightly improved version called a weak design. We want a set system of large cardinality but with pairwise small intersections. The Nisan-Widgerson design requires that this intersection being small for all pairs, the RRV weak design only requires this in an aggregate sense.

More formally,

Definition 6.2 *For $\ell \in \mathbf{N}, \rho \geq 1$, an (ℓ, ρ)-design is a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ with $S_i \subset [d]$ such that:*

- *$\forall i, |S_i| = \ell$, and*
- *$\forall i \neq j, |S_i \cap S_j| \leq \log \rho$.*

The Nisan-Widgerson design uses low degree polynomials as follows. Let q be a prime power, $d = q^2$, then each set S_i is determined by a polynomial f of some degree bound D, such that $S_i = \{(a, f(a)) \mid a \in \mathbf{F}_q\}$. Then each $|S_i| = q$ and $S_i \subset [d]$. Since distinct polynomials of degree at most D can agree at most D points, $|S_i \cap S_j| \leq D$, for $i \neq j$. The astute reader will recognize that this is a generalization of the projective plane (here actually an affine plane) over a finite field. (In the projective plane, we use linear polynomials. Here the polynomials f are of degree bounded by some upper bound D.)

A weak design is defined as follows:

Definition 6.3 *For $\ell \in \mathbf{N}, \rho \geq 1$, a weak (ℓ, ρ)-design is a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ with $S_i \subset [d]$ such that:*

- *$\forall i, |S_i| = \ell$, and*
- *$\forall i, \sum_{j<i} 2^{|S_i \cap S_j|} \leq \rho(m-1)$.*

Note that if $\mathcal{S}$ is a (ℓ, ρ)-design, then it is also a weak (ℓ, ρ)-design.

We are now ready to describe the Trevisan and RRV extractor. Let $m \leq k \leq n$, and $\epsilon > 0$. It takes an imperfect random source of n bits with min-entropy k as well as d uniform bits, and outputs m bits that are ϵ-close to uniform. Let $\delta = \epsilon/4m$, and consider the error correcting code $\mathrm{EC}_{n,\delta}$. [1] Define $\ell = \log_2 \overline{n}$ and let $\mathcal{S}$ be a weak (ℓ, ρ)-design. We will first apply $\mathrm{EC}_{n,\delta}$ to the imperfect random bits to generate $\overline{n}$ bits. Then these $\overline{n}$ bits is viewed as a truth table for a Boolean function $\overline{u}$ on ℓ bits. We will then apply the function $\overline{u}$ to different subsets of the d random bits, choosing which subsets using the design $\mathcal{S}$, and this will be the output of the generator. (This last part of the extractor—choosing different subsets using a design, and then evaluating a function at these subsets of bits—is just the Nisan-Wigderson generator using $\overline{u}$ as the hard function with seed $y \in_U \{0,1\}^d$.)

$$\begin{aligned} \mathrm{Ext}_{\mathcal{S}} : \{0,1\}^n \times \{0,1\}^d &\rightarrow \{0,1\}^m \\ \mathrm{Ext}_{\mathcal{S}}(u, y) = \mathrm{NW}_{\mathcal{S},\overline{u}}(y) &= \overline{u}(y|_{S_1}) \cdots \overline{u}(y|_{S_m}) \end{aligned}$$

where $\overline{u} = \mathrm{EC}_{n,\delta}(u)$.

We now outline a proof that $\mathrm{Ext}_{\mathcal{S}}$ is an extractor. In fact it is a strong extractor, i.e., even with the additional d random bits present, $(U_d, \mathrm{Ext}_{\mathcal{S}}(X, U_d))$ is ϵ-close to the uniform distribution U_{m+d} for any X with $H_\infty(X) \geq k$.

We prove by contradiction. Suppose this is not true, then for some distribution X on $\{0,1\}^n$ with $H_\infty(X) \geq k$, the output of $(U_d, \mathrm{Ext}_{\mathcal{S}})$ can be distinguished from the uniform distribution by $> \epsilon$. Then by a classical argument due to Yao, using an interpolation of distributions, one can show for some i, there exists a next-bit predictor $A : \{0,1\}^d \times \{0,1\}^{i-1} \rightarrow \{0,1\}$,

$$\Pr_{y \in_U \{0,1\}^d, u \leftarrow X}[A(y, \overline{u}(y|_{S_1}), \ldots, \overline{u}(y|_{S_{i-1}})) = \overline{u}(y|_{S_i})] > \frac{1}{2} + \frac{\epsilon}{m}.$$

In other words, when given $y \in_U \{0,1\}^d$ and $\overline{u}(y|_{S_1}), \ldots, \overline{u}(y|_{S_{i-1}})$, A will provide the correct value of $\overline{u}(y|_{S_i})$ with probability $> 1/2 + \epsilon/m$.

Consider the set B of "bad" u, for which A provides a good prediction to $\overline{u}(y|_{S_i})$,

$$\Pr_{y \in_U \{0,1\}^d}[A(y, \overline{u}(y|_{S_1}), \ldots, \overline{u}(y|_{S_{i-1}})) = \overline{u}(y|_{S_i})] > \frac{1}{2} + \frac{\epsilon}{2m}.$$

For any u, let

$$p = \Pr_y[A(y, \overline{u}(y|_{S_1}), \ldots, \overline{u}(y|_{S_{i-1}})) = \overline{u}(y|_{S_i})].$$

If we focus on the $\ell = |S_i|$ bits within S_i, by an averaging argument, we can set all the other bits $\{0,1\}^d - S_i$ to some particular 0-1 assignment, and preserve this probability p.

$$\Pr_x[A(y(x), \overline{u}(y(x)|_{S_1}), \ldots, \overline{u}(y(x)|_{S_{i-1}})) = \overline{u}(x)] \geq p,$$

where $x = y|_{S_i}$, and $y(x)$ is y with all the bits in $\{0,1\}^d - S_i$ already set to some 0-1 constants. Now, each function $\overline{u}(y(x)|_{S_j})$, for $j < i$, is a Boolean function on $S_i \cap S_j$, which can be specified by a truth table of length $2^{|S_i \cap S_j|}$, Therefore the set of sequences of Boolean functions $x \mapsto (y(x), \overline{u}(y(x)|_{S_1}), \ldots, \overline{u}(y(x)|_{S_{i-1}}))$ can be counted by a binary string of length $\sum_{j<i} 2^{|S_i \cap S_j|} \leq \rho(m-1)$. Let $\mathcal{F}$ denote the set of all such sequences of Boolean functions. Then $|\mathcal{F}| < 2^{\rho m}$. This is a consequence of the weak-design $\mathcal{S}$.

Then, each $u \in B$ satisfies the following property: There exists some $f \in \mathcal{F}$, such that

$$\Pr_x[A(f(x)) = \overline{u}(x)] = \Pr_x[A(y(x), \overline{u}(y(x)|_{S_1}), \ldots, \overline{u}(y(x)|_{S_{i-1}})) = \overline{u}(x)] \geq p > \frac{1}{2} + \frac{\epsilon}{2m}.$$

[1] The output size of the extractor is customarily denoted as m, and is unrelated to the field size chosen for $\mathrm{EC}_{n,\delta}$, which is $\approx n/\delta^2$.

But we can view the Boolean function $A(f(\cdot))$ as a binary string, via its truth table, of length $\overline{n}$, and then the probability $\Pr_x[A(f(x)) \neq \overline{u}(x)]$ has another interpretation, namely the relative Hamming distance between these two strings in $\{0,1\}^{\overline{n}}$. Thus $\Pr_x[A(f(x)) = \overline{u}(x)] > \frac{1}{2} + \frac{\epsilon}{2m}$ is the same as relative Hamming distance $< \frac{1}{2} - \frac{\epsilon}{2m}$.

Now recall our $\overline{u}$ is the output of our error correcting code $\mathrm{EC}_{n,\delta}$, it follows that within the relative Hamming distance $\frac{1}{2} - \frac{\epsilon}{2m}$ of $A(f(\cdot))$ there can be at most $(2m/\epsilon)^2$ strings $\overline{u}$. This is a consequence of the error correcting code $\mathrm{EC}_{n,\delta}$.

Thus

$$|B| \leq (2m/\epsilon)^2 |\mathcal{F}| < (2m/\epsilon)^2 2^{\rho m}.$$

It is possible to have a weak-design $\mathcal{S}$ with $\rho = (k - 3\log_2(m/\epsilon) - 3)/m$. Since X has min-entropy k, it follows that

$$\Pr_{u \leftarrow X}[u \in B] < (2m/\epsilon)^2 2^{\rho m} \cdot 2^{-k} = \epsilon/2m.$$

On the other hand, if $u \notin B$, then by definition

$$\Pr_{y \in_U \{0,1\}^d}[A(y, \overline{u}(y|_{S_1}), \ldots, \overline{u}(y|_{S_{i-1}})) = \overline{u}(y|_{S_i})] \leq \frac{1}{2} + \frac{\epsilon}{2m}.$$

Combining,

$$\begin{aligned}
\Pr_{y \in_U \{0,1\}^d, u \leftarrow X}[A(y, \overline{u}(y|_{S_1}), \ldots, \overline{u}(y|_{S_{i-1}})) = \overline{u}(y|_{S_i})] &\leq \Pr_{u \leftarrow X}[u \in B] + \Pr_{u \leftarrow X}[u \notin B]\left(\frac{1}{2} + \frac{\epsilon}{2m}\right) \\
&\leq \frac{\epsilon}{2m} + \left(\frac{1}{2} + \frac{\epsilon}{2m}\right) \\
&= \frac{1}{2} + \frac{\epsilon}{m},
\end{aligned}$$

This contradicts the performance of the predictor A.

Theorem 6.1 *$\forall n, k, m \in \mathbf{N}, \epsilon > 0, m \leq k \leq n$, there exists an explicit (n, d, k, m, ϵ)-extractor with $d = O\left(\frac{log^2(\frac{n}{\epsilon})}{log(\frac{k}{m})}\right)$.*

7 Valiant's New Theory of Holographic Algorithms

In a remarkable paper, Valiant [69] in 2004 has proposed a completely new theory of Holographic Algorithms or Holographic Reductions. The theory is quite unlike anything before that it is probably fair to say that at this point few people in the world understands or appreciates its full reach and potentials. (We are certainly not among those.)

It is a delicate theory that will be difficult to explain without all the definitions. At the heart of the new theory, one can say that Valiant has succeeded in devising a custom made process which is capable of carrying out a seemingly exponential computation with exponentially many cancellations so that the computation can actually be done in polynomial time. One can view this new theory as another algorithmic design paradigm, much like greedy algorithms, or divide-and-conquer, or dynamic programming etc, which pushes back the frontier of what is solvable by polynomial time. Admittedly, at this early stage, it is still premature to say what drastic consequence it might have on the landscape of the big questions of complexity theory, such as P vs. NP. But the new theory has already been used by Valiant to devise polynomial time algorithms for a number of problems. All these problems appear to be very close to NP-complete or NP-hard problems in one way or another. All of which were thought to have no known polynomial time algorithms, and all appear to need the new machinery to show their membership in P. [2]

Unless and until a proof of P $\neq$ NP is found, one should regard this conjecture as just that, an unproven conjecture. We can ask ourselves on what basis we derive confidence on the truth of this conjecture. In our view, frankly, it is not for any partial lower bound. While generally ingenious and difficult to prove, all lower bounds we currently have are either for very restricted models of computation or are still very weak. Fundamentally this source of confidence in P $\neq$ NP comes from the fact that all existing algorithmic approaches do not seem to tackle these NP-complete problems. Well, when there is a new algorithmic design paradigm being found, one must re-examine all these beliefs critically.

We now give a brief description of this theory. Playing what seems to be a fundamental role is the planar matching problem. Given a graph G, a matching of G is a set of edges no two of which share a vertex. A perfect matching M is a matching such that every vertex of G is incident to one edge of M. The decision problem of whether there is a perfect matching in G is computable in P, one of the notable achievements in the study of Algorithms. However, it is known that counting the number of perfect matchings in G is #P-complete.

We assign to every edge $e = (i, j)$ a variable x_{ij}, where $i < j$. Then we define the following polynomial

$$\mathrm{PerfMatch}(G) = \sum_{M} \prod_{(i,j) \in M} x_{ij},$$

where the sum is over all perfect matchings M. $\mathrm{PerfMatch}(G)$ is a polynomial on $\binom{n}{2}$ many variables x_{ij}, where $1 \leq i < j \leq n$. If the graph is a weighted graph with weights w_{ij}, we can also evaluate $\mathrm{PerfMatch}(G)$ at $x_{ij} = w_{ij}$. Note that if all the weights are 1, then $\mathrm{PerfMatch}(G)$ just counts the number of perfect matchings in the graph.

A most remarkable result due to Fisher, Kasteleyn and Temperly (FKT), ([64], [35], and [36]) from statistical physics is that for planar graphs, this Perfect Matching polynomial $\mathrm{PerfMatch}(G)$ can be evaluated in polynomial time. In fact it can be evaluated as a Pfaffian of a skew-symmetric matrix which is constructible from a planar embedding of G in polynomial time, (and a planar embedding of G can also be computed from a planar graph G given by its adjacency matrix in polynomial time).

In effect, all of Valiant's successes on these problems for which he found P time algorithms for the first time are obtained by a certain reduction to this planar Perfect Matching polynomial evaluation. These reductions are called holographic reductions, because they carry out exponentially many cancellations in a pattern of interference that is "tailor made" for the problem.

Consider a graph which is a line segment with 4 vertices $\{1, 2, 3, 4\}$ and 3 edges $\{(1, 2), (2, 3), (3, 4)\}$ with weights -1, 1 and 1 respectively. The $\mathrm{PerfMatch}(G)$ is -1, as $\{(1, 2), (3, 4)\}$ is the unique perfect matching. If we remove either the left or the right end vertex then there are no more perfect matchings and therefore $\mathrm{PerfMatch}(G - \{1\}) = \mathrm{PerfMatch}(G - \{4\}) = 0$. If we remove both end vertices then we have a unique perfect matching again $\{(2, 3)\}$ and $\mathrm{PerfMatch}(G - \{1, 4\}) = 1$.

Generally we define a planar matchgate Γ as a triple (G, X, Y) where G is a planar embedding of a weighted planar graph (V, E, W), where $X \subseteq V$ is a set of input nodes, $Y \subseteq V$ is a set of output nodes, and $X \cap Y = \emptyset$. Furthermore in the planar embedding of G, counter-clock wise one encounters vertices of X, labeled $1, \ldots, |X|$ and then vertices of Y, labeled $|Y|, \ldots, 1$. Now the

[2] In a most recent version of [69], available in ECCC, Valiant has found that for a few problems on that list, there is an alternative approach using classical reductions from a problem called #PL-CUT. This works with respect to a particular basis called **b2**. But even for this basis, macthgates of arity 4 still appear to be out of reach by classical reductions.

standard signature, $u(\Gamma)$, of Γ is a $2^{|X|} \times 2^{|Y|}$ matrix whose entries are indexed by subsets $X' \subseteq X$ and $Y' \subseteq Y$, and the entry indexed by (X', Y') is PerfMatch$(G - Z)$, where $Z = X' \cup Y'$.

Matchgates with only output nodes are called generators. Matchgates with only input nodes are called recognizers. More generally, with both input and output nodes a matchgate is called a transducer. The above example of a line segment of 4 nodes, with nodes $1, 4$ designated as output nodes and no input nodes is a generator. Its standard signature $u(\Gamma)$ is a 2^0 by 2^2 matrix (ie a 4-dimensional vector) $(-1, 0, 0, 1)$.

Let $\mathbf{b}_0$ denote the standard basis for two dimensional space $\mathbf{b}_0 = [e_0, e_1] = [(1, 0), (0, 1)]$. Consider another basis $\mathbf{b}_1 = [n, p] = [(-1, 1), (1, 0)]$. Clearly $p = e_0$, and $n + p = e_1$. Note that the tensor product $(n + p) \otimes (n + p) = n \otimes n + n \otimes p + p \otimes n + p \otimes p$. The above standard signature $u(\Gamma)$ is

$$(-1, 0, 0, 1) = e_1 \otimes e_1 - e_0 \otimes e_0 = (n + p) \otimes (n + p) - p \otimes p = n \otimes n + n \otimes p + p \otimes n.$$

This expression in Valiant's theory is an expression of a "superposition" of 3 "boolean bit patterns" 00, 01, 10.

In general Valiant's theory considers any basis $\mathbf{b}$ consisting of vectors of length 2^a for some arity a. In the following we will assume $a = 1$ as above, and we denote by $\mathbf{b} = [n, p]$. Let T be the basis transformation matrix from $\mathbf{b}_0$ to $\mathbf{b}$. In the case of the transformation from $\mathbf{b}_0$ to $\mathbf{b}_1$, we have $T = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix}$.

Now Valiant defines the objects called valG and valR.

For any generator Γ with k output nodes, and any vector $x \in \{n, p\}^{\otimes k}$, valG$(\Gamma, x)$ is the coefficient of x when we express the standard signature $u(\Gamma)$ in the basis of $\{n, p\}^{\otimes k}$.

This can be defined equivalently as follows. We form the tensor product matrix $T^{\otimes k}$ which transforms the basis $(\mathbf{b}_0)^{\otimes k}$ to $(\mathbf{b})^{\otimes k}$. Then valG$(\Gamma, \cdot)$ is obtained from the standard signature $u(\Gamma)$ by multiplying the tensor product matrix $(T^{-1})^{\otimes k} = (T^{\otimes k})^{-1}$:

$$\text{valG}(\Gamma, \cdot) = u(\Gamma)(T^{\otimes k})^{-1},$$

where for a generator Γ, $u(\Gamma)$ is a row vector of dimension 2^k.

Similarly for any recognizer Γ' with k input nodes, and any vector $x \in \{n, p\}^{\otimes k}$, Valiant defines valR$(\Gamma', x)$ by first transforming x to the standard basis $(\mathbf{b}_0)^{\otimes k}$ and then taking inner product of this with the standard signature $u(\Gamma')$.

Equivalently and more succinctly, in our linear algebra notation

$$\text{valR}(\Gamma', \cdot) = T^{\otimes k} u(\Gamma'),$$

where for a recognizer Γ', $u(\Gamma')$ is a column vector.

In the above definitions of valG and valR, we have presented Valiant's definitions, followed by a vector form which is equivalent to his definitions. This will hopefully help clarify the following presentations. As argued in an upcoming paper by Cai and Choudhary [15], an even more appropriate home for these objects valG and valR are in various *covariant* and *contravariant* tensor spaces. But for this article, we will go no further than the vector formulation.

A matchgrid Ω is a weighted planar graph G consisting of a disjoint union of: a set of g generators $A = (A_1, \ldots, A_g)$, a set of r recognizers $B = (B_1, \ldots, B_r)$, and a set of f connecting edges $C = (C_1, \ldots, C_f)$, where each C_i has weight 1 and connects exactly one output node of some A_i to one input node of some B_j, in such a way that all the input and output nodes are matched.

Now we come to the central definition of Valiant's theory—the Holant.

$$\text{Holant}(\Omega) = \sum_{x \in \mathbf{b}^{\otimes f}} \left\{ [\Pi_{1 \le i \le g} \text{valG}(A_i, x|_{A_i})] \cdot [\Pi_{1 \le j \le r} \text{valR}(B_j, x|_{B_j})] \right\}.$$

The following theorem is the beautiful *Holant Theorem* of Valiant. This theorem is the linchpin that holds everything together and makes it all possible for holographic algorithms.

Theorem 7.1 (Valiant) *For any matchgrid Ω over any basis $\mathbf{b}$, let G be its underlying weighted graph, then*

$$\text{Holant}(\Omega) = \text{PerfMatch}(G).$$

We will sketch a proof. Hopefully the definitions of valG and valG in our linear algebra expressions will make the proof of this amazing theorem more tractable. The key observation we make is that in the definition of Holant, we are in fact evaluating an inner product of two vectors, the first of which is the tensor product of all the $\text{valG}(A_i, \cdot)$ over the generators, and the second is the tensor product of all the $\text{valG}(B_j, \cdot)$ over the recognizers. Thus

$$\text{Holant}(\Omega) = \langle \bigotimes_i \text{valG}(A_i, \cdot), \bigotimes_j \text{valR}(B_j, \cdot) \rangle.$$

Note that the sum $\sum_{x \in \mathbf{b}^{\otimes f}}$ is precisely over all the entries in the two tensor product vectors indexed by $x \in \mathbf{b}^{\otimes f}$.

Let A_i have k_i output nodes. We make the observation that the tensor product $\bigotimes_i \text{valG}(A_i, \cdot)$ can be expressed by the vector-matrix product

$$[\otimes_i u(A_i)][\otimes_i (T^{\otimes k_i})^{-1}],$$

according to an appropriate ordering of the indices, which is just $[\otimes_i u(A_i)](T^{\otimes \sum_i k_i})^{-1}$.

Similarly the tensor product $\bigotimes_j \text{valR}(B_j, \cdot)$ can be expressed by

$$T^{\otimes \sum_j \ell_j}[\otimes_j u(B_j)],$$

as a column vector, where each recognizer B_j has ℓ_j input nodes. And of course by the definition of a matchgrid, $\sum_i k_i = \sum_j \ell_j = f$, the number of interconnecting wires between the generators and the recognizers. Finally we get

$$\text{Holant}(\Omega) = \langle [\otimes_i u(A_i)](T^{\otimes f})^{-1}, T^{\otimes f}[\otimes_j u(B_j)] \rangle.$$

Now the beautiful thing is that the adjacent $T^{\otimes f}$ and $(T^{\otimes f})^{-1}$ cancel. What we have left is the definition of the Holant under the standard basis $\mathbf{b}_0$,

$$\text{Holant}(\Omega) = \langle \otimes_i u(A_i), \otimes_j u(B_j) \rangle.$$

Now the equality of

$$\text{Holant}(\Omega) = \text{PerfMatch}(G)$$

becomes clearer: If we evaluate the $\text{PerfMatch}(G)$ polynomial by classifying all the perfect matchings M according to what subset S of edges among f connecting wires $\{C_1, \ldots, C_f\}$ it contains, then it is not difficult to see that corresponding to all perfect matchings M with $M \cap \{C_1, \ldots, C_f\} = S$, we get exactly the term

$$[\Pi_{1 \le i \le g} \text{PerfMatch}(A_i - S|_{A_i})] \cdot [\Pi_{1 \le j \le r} \text{PerfMatch}(B_j - S|_{B_j})],$$

where $S|_{A_i}$ denotes the subset of output nodes of A_i incident to S, and similarly for $S|_{B_j}$. This is exactly the corresponding term in the inner product of

$$\langle \otimes_i u(A_i), \otimes_j u(B_j) \rangle,$$

proving the Theorem.

More generally, we can define a matchgrid Ω consisting of generators, recognizers as well as transducers.

Let Γ be a transducer with ℓ input nodes and k output nodes. The standard signature $u(\Gamma)$ has already been defined, which is a $2^\ell \times 2^k$ matrix. We define the object $\mathrm{valT}(\Gamma, \cdot)$ under basis $\mathbf{b}$ to be a $2^\ell \times 2^k$ matrix

$$T^{\otimes \ell} u(\Gamma) (T^{\otimes k})^{-1}.$$

We define matchgrid Ω to be a weighted planar graph G which consists of a disjoint set of g generators $A_1, \ldots, A_g$, r recognizers $B_1, \ldots, B_r$, t transducers $\Gamma_1, \ldots, \Gamma_t$, and f connecting edges $C_1, \ldots, C_f$, where each C_i has weight 1 and they connect output nodes of some A_α or Γ_γ to input nodes of some B_β or $\Gamma_{\gamma'}$ in a 1-1 fashion.

Then we can define the extended Holant:

$$\mathrm{Holant}(\Omega) = \sum_{x \in \mathbf{b}^{\otimes f}} \left\{ [\Pi_{1 \le \alpha \le g} \mathrm{valG}(A_\alpha, x|_{A_\alpha})] \cdot [\Pi_{1 \le \beta \le r} \mathrm{valR}(B_\beta, x|_{B_\beta})] \cdot [\Pi_{1 \le \gamma \le t} \mathrm{valT}(\Gamma_\gamma, x|_{\Gamma_\gamma})] \right\}.$$

Again we can prove Valiant's Holant Theorem

Theorem 7.2 *For any matchgrid Ω with generators, recognizers and transducers, over any basis* $\mathbf{b}$*, let G be its underlying weighted graph, then*

$$\mathrm{Holant}(\Omega) = \mathrm{PerfMatch}(G).$$

Encapsulated in Valiant's theorem is a custom-made exponential set of cancellations. And the computation is ultimately reduced to that of computing the Perfect Matching polynomial $\mathrm{PerfMatch}(G)$, which, by the FKT method can be computed in polynomial time for planar graphs G.

For expository purposes we have assumed the basis $\mathbf{b}$ to consist of 2 linearly independent vectors (for which we have a square invertible matrix T.) But strictly speaking this is not necessary. As long as the standard signature $u(\Gamma)$ is in the linear span of the tensor products of the basis $\mathbf{b}$,

$$u(\Gamma) \in \mathrm{span}(\mathbf{b}^{\otimes k}),$$

we can define $\mathrm{valG}(\Gamma, \cdot)$ as any vector v such that

$$u(\Gamma) = v T^{\otimes k},$$

then the Holant Theorem above still holds with the same proof.

In particular, as long as the linear span of $\mathbf{b}$ is the same as that of $\mathbf{b_0}$, this holds. This is because in this case $\mathrm{span}(\mathbf{b}^{\otimes k})$ has full dimension 2^k.

A number of counting problems (and decision problems), which did not have any known polynomial time algorithms, were shown to be solvable in polynomial time due to this magic formula by Valiant.

We mention three such problems. For the first problem the input is a planar graph of maximum degree 3. We want to compute the number of orientations such that no node has all the edges directed towards it or away from it. Here an orientation of an undirected graph is to assign for each edge exactly one direction.

For the second problem the input is a planar graph of maximum degree 3. We want to compute the cardinality of a smallest subset of vertices such that its removal renders the graph bipartite. This problem is NP-complete for maximum degree 6.

The third problem is called #X-Matchings. After its description we will describe its solution using the Holant Theorem. The solutions to the other two problems as well as several others can be found in [69].

The input to #X-Matchings is a planar weighted bipartite graph $G = (V, E, W)$ where V is partitioned into V_1 and V_2, and every node in V_1 has degree 2. The output is the sum of the masses of all matchings of all sizes where the mass of a matching is the product of (1) the weights of all the edges present in the matching, and (2) the quantity $-(w_1 + \ldots + w_k)$ for all nodes in V_2 which are unmatched, where $w_1, \ldots, w_k$ are all the weights of edges incident to that unmatched node.

Now we will define a matchgrid.

For this purpose we will consider the basis $\mathbf{b}_1 = [n, p]$, and the following gadgets for generators A_i and recognizers B_j. Each vertex u_i of V_1 is replaced by a generator A_i, and each A_i is the same gadget. This gadget is the line graph with 4 vertices described as an example earlier illustrating the Perfect Matching polynomial PerfMatch(G). Its standard signature $u(A_i) = (-1, 0, 0, 1) = n \otimes n + n \otimes p + p \otimes n$. One can say that this generator generates the "bit pattern" (nn, np, pn, pp) with coefficients $(1, 1, 1, 0)$ under the basis $\mathbf{b}_1$. If we think of this corresponds to a "logical bit pattern", with n for no and p for yes, then this corresponds to allowing having zero or one (either one) but not both edges at u_i being picked. Such a scenario naturally corresponds to a (not necessarily perfect) matching at the vertex $u_i \in V_1$.

The recognizer gadget for each vertex v in V_2 of degree d is a "star graph" B with one vertex v_0 at the center and d many vertices outside, each connected to v_0 by an edge: The vertex set of B is $\{v_0, v_1, \ldots, v_d\}$, and edge set is $\{(v_0, v_1), \ldots, (v_0, v_d)\}$, where each edge (v_0, v_i) inherits the weight in the input graph for the ith edge at v. If the ith edge at v is connected to $u \in V_1$, then v_i is connected to an output node of the gadget A for u, and with weight 1. These $2|V_1| = \sum_{v \in V_2} \deg(v)$ edges are the f connecting edges described in the general matchgrid construction.

It is clear that this local replacement procedure starting from the planar graph G produces still a planar graph, which is our matchgrid.

For the "star graph" B as a recognizer with d outside vertices as its input nodes, the standard signature $u(B)$ has dimension 2^d. Every subset $Z \subseteq \{1, \ldots, d\}$ produces no perfect matchings in $B - Z$, except those Z which has cardinality exactly $d - 1$, and in which case the entry in $u(B)$ is w_i.

Now consider valR$(B, \cdot)$. It is not difficult to show that valR$(B, \cdot)$ is a vector with most entries 0, except those entries labelled by $\mathbf{x} = x_1 \otimes x_2 \otimes \ldots \otimes x_d$ with either all, or all except one, $x_i = n$. If $x_i = p$ for one i, and all the rest are n, then the value of valR is w_i, where w_i is the weight of the ith edge at $v \in V_2$. If all $x_i = n$, then the value of this entry valR$(B, n \otimes n \otimes \ldots \otimes n) = -\sum_{i=1}^{d} w_i$.

We can see this as follows: Suppose some $\mathbf{x} = x_1 \otimes x_2 \otimes \ldots \otimes x_d$, where $x_i = n = (-1, 1)$ or $x_i = p = (1, 0)$, has a non-zero inner product with $u(B)$. We only need to focus on those entries of $u(B)$ indexed by a subset Z of cardinality exactly $d - 1$. At an arbitrary entry indexed by Z, the value in the vector $\mathbf{x}$ is a product of the second or the first entries of x_i, depending on $i \in Z$ or not. Thus if $|Z| = d - 1$, we take the product from the first entry for one x_i, and the second entry from the other x_i's for exactly $d - 1$ times. If there are more than one $x_i = p$, then for each $|Z| = d - 1$, for at least one $i \in Z$ we have $x_i = p$, and the corresponding factor is the second entry 0 in $p = (1, 0)$, giving the product value 0. For those $\mathbf{x}$ with exactly zero or one $x_i = p$ the valR$(B, \cdot)$ values are clearly as stated.

Again by a "bit pattern" interpretation with n for no and p for yes, at a vertex $v \in V_2$, the only non-zero valR$(B, \mathbf{x})$ values are those having zero or one $x_i = p$ corresponds to (not necessarily perfect) matchings.

Now the Holant definition expresses exactly the sum to be computed by the problem #X-Matchings. On the other hand, the Holant Theorem shows that the Holant can be computed by evaluating the PerfMatch polynomial on the matchgrid via the FKT method in polynomial time.

It was shown by Jerrum [32] that the problem of counting the number of (not necessarily perfect) matchings in a planar graph is #P-complete. Vadhan [66] showed that this was true even for planar bipartite graphs of degree 6. If all vertices have degree 2, the problem is trivial. Therefore the above polynomial time algorithm for #X-Matchings is quite remarkable. Note that if all vertices in V_2 have degree 4 and we set all weights to 1, and evaluate everything modulo 5, then $-\sum_{i=1}^{d} w_i = -4 \equiv 1 \bmod 5$. Then we would be evaluating the number of (not necessarily perfect) matchings in the planar bipartite graph modulo 5, in polynomial time.

Valiant's Theory of Holographic Algorithms were developed along with a closely related theory of matchgates in terms of Pfaffians and simulations of certain quantum computations [68, 67, 70, 71]. In the Pfaffian based development, one drops the planarity requirement, but adds a complication of order-overlaps; in this parallel theory, instead of *signatures* one has the notion of *characters*. It appears that there is a great deal of depth and richness to this theory. For instance, these matchgates satisfy a set of matchgate identities which follow from the Grassmann-Plücker identities. Valiant showed a set of 5 identities to be satisfied by 2-input 2-output matchgates. Cai, Choudhary and Kumar have found a complete set of 10 identities characterizing these matchgates [16]. It turns out that, using these identities and *Jacobi's Theorem* on *determinantal compounds*, we can show that the invertible characters form a group. We can also show a certain equivalence of the two theories. Many structural properties and the limit of these computations are still open.

Acknowledgments

We would like to thank Leslie Valiant and Andrew Yao for helpful comments. We also thank Vinay Choudhary, Rakesh Kumar and Anand Sinha for many interesting discussions, and to Vinay Choudhary for his help in the bibliography.

References

[1] L. Adleman and M.-D. Huang. Recognizing primes in random polynomial time. In *Proc. 19th ACM Symposium on Theory of Computing*, 1987, 462–469.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. Annals of Mathematics, 160: 781–793 (2004).

[3] Dorit Aharonov and Oded Regev. Lattice Problems in NP $\cap$ coNP. In *Proc. 45th IEEE Symposium on Foundations of Computer Science*, 2004, 362–371.

[4] M. Ajtai. Generating hard instances of lattice problems. In *Proc. 28th ACM Symposium on the Theory of Computing*, 1996, 99–108. Full version available from ECCC as TR96-007.

[5] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proc. 30th ACM Symposium on the Theory of Computing*, 1998, 10–19. Full version available from ECCC as TR97-047.

[6] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. 29th ACM Symposium on the Theory of Computing*, 1997, 284–293. Full version available from ECCC as TR96-065.

[7] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems. In *Proc. 20th IEEE Symposium on Foundations of Computer Science*, 1979, 218–223.

[8] R. Armoni, Amnon Ta-Shma, A. Wigderson, and S. Zhou. An $O(\log(n)^{4/3})$ space algorithm for (s,t) connectivity in undirected graphs. *Journal of the ACM*, 47(2): 294–311 (2000).

[9] László Babai. Trading Group Theory for Randomness. In *Proc. 17th ACM Symposium on Theory of Computing*, 1985, 421–429

[10] László Babai, Shlomo Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. *Journal of Computer and System Sciences*, 36(2): 254-276 (1988).

[11] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, Avi Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. In *Proc. 20th ACM Symposium on Theory of Computing*, 1988, 113–131.

[12] Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudo Random Bits. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science*, 1982, 112–117.

[13] J-Y. Cai. A Worst-Case to Average-Case Connection for Closest Vector Problem. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science*, 2001, 308–317.

[14] J-Y. Cai, V. T. Chakaravarthy, D. van Melkebeek. Time-Space Tradeoff in Derandomizing Probabilistic Logspace. In *Proc. 21st International Symposium on Theoretical Aspects of Computer Science*, 2004, 571–583.

[15] J-Y. Cai and V. Choudhary. Valiant's Holant Theorem and Matchgate Tensors. Available from ECCC, *Electronic Colloquium on Computational Complexity* TR05-118, at `http://www.eccc.uni-trier.de/eccc/`.

[16] J-Y. Cai, V. Choudhary and R. Kumar. On Valiant's holographic algorithms. To appear.

[17] J-Y. Cai and A. Nerurkar. An Improved Worst-Case to Average-Case Connection for Lattice Problems. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, 1997, 468–477.

[18] J-Y. Cai. A Relation of Primal-Dual Lattices and the Complexity of Shortest Lattice Vector Problem. *Theoretical Computer Science* 207:105–116, 1998.

[19] S. A. Cook. The complexity of theorem proving procedures. In *Proc. 3rd ACM Symposium on Theory of Computing*, 1971, 151–158.

[20] L. Carter and M. Wegman. Universal Hash Functions. *Journal of Computer and System Sciences*, 18: 143–154 (1979).

[21] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms* (Second Edition). MIT Press and McGraw-Hill, 2001.

[22] A. Granville, It Is Easy to Determine Whether a Given Integer Is Prime. Bull. Amer. Math. Soc. 42, 3-38, 2005.

[23] Irit Dinur. The PCP theorem by gap amplification. *Electronic Colloquium on Computational Complexity*, Report TR05-046. http://eccc.uni-trier.de/eccc-reports/2005/TR05-046/index.html

[24] Uriel Feige, Shafi Goldwasser, Lszl Lovsz, Shmuel Safra, Mario Szegedy. Interactive Proofs and the Hardness of Approximating Cliques. *Journal of the ACM*, 43(2): 268–292 (1996).

[25] O. Goldreich and S. Goldwasser. On the Limits of Non-Approximability of Lattice Problems. In *Proc. 30th ACM Symposium on Theory of Computing*, 1998, 1–9. J. Comput. Syst. Sci. 60(3): 540-563 (2000).

[26] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology – CRYPTO '97*, Burton S. Kaliski Jr. (Ed.), Lecture Notes in Computer Science, 1294: 112-131, Springer-Verlag, 1997.

[27] Shafi Goldwasser, Silvio Micali, Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, 18(1): 186–208 (1989).

[28] Shafi Goldwasser, Michael Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In *Proc. 18th ACM Symposium on Theory of Computing*, 1986, 59–68.

[29] J. Hastad, R. Impagliazzo, L. Levin and M. Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, 28(4): 1364–1396 (1999).

[30] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.

[31] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5): 935–938 (1988).

[32] M. Jerrum. Two-dimensional monomer-dimer systems are computationally intractible, J. Stat. Phys. 48 (1987) 121-134; erratum, 59 (1990) 1087-1088.

[33] Richard Karp. Reducibility Among Combinatorial Problems. In *Proc. of a Symposium on the Complexity of Computer Computations*, 1972. In Miller, R. E. and Thatcher, J. W., editors, Complexity of Computer Computations, pages 85–103. Plenum Press, New York, NY.

[34] Subhash Khot. Hardness of Approximating the Shortest Vector Problem in Lattices. In *Proc. 45th IEEE Symposium on Foundations of Computer Science*, 2004, 126–135.

[35] P. W. Kasteleyn. The statistics of dimers on a lattice. *Physica*, 27: 1209-1225 (1961).

[36] P. W. Kasteleyn. Graph Theory and Crystal Physics. In *Graph Theory and Theoretical Physics*, (F. Harary, ed.), Academic Press, London, 43-110 (1967).

[37] Donald E. Knuth. *The Art of Computer Programming*, Third Edition. Reading, Massachusetts: Addison-Wesley, 1997.

[38] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261: 515–534, 1982.

[39] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[40] L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity.* SIAM, Philadelphia, 1986.

[41] Chi-Jen Lu, Omer Reingold, Salil P. Vadhan, Avi Wigderson. Extractors: optimal up to constant factors. In *Proc. 35th ACM Symposium on the Theory of Computing*, 2003, 602–611.

[42] Carsten Lund, Lance Fortnow, Howard J. Karloff, Noam Nisan. Algebraic Methods for Interactive Proof Systems. In *Proc. 31st IEEE Symposium on Foundations of Computer Science*, 1990, 2–10.

[43] N. Metropolis and S. Ulam The Monte Carlo Method. *Journal of American Statistical Association*, 44:335–341 (1949).

[44] D. Micciancio. The Shortest Vector in a Lattice is Hard to Approximate to within Some Constant. In *Proc. 39th IEEE Symposium on Foundations of Computer Science*, 1998, 92–98.

[45] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. In *Proc. 45th IEEE Symposium on Foundations of Computer Science*, 2004, 372–381.

[46] G. L. Miller. Riemann's hypothesis and tests for primality. In *Proc. Seventh Annual ACM Symposium on the Theory of Computing*, 1975, 234–239.

[47] P. Nguyen and J. Stern. A converse to the Ajtai-Dwork security proof and its cryptographic implications. Available from ECCC as TR98-010.

[48] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4): 449–461 (1992).

[49] Noam Nisan. $RL \subseteq SC$. In *Proc. 24th ACM Symposium on the Theory of Computing*, 1992, 619–623.

[50] Noam Nisan, Endre Szemerédi, and Avi Wigderson. Undirected connectivity in $O(\log^{1.5} n)$ space. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, 1992, 24–29.

[51] Noam Nisan, David Zuckerman. Randomness is Linear in Space. *Journal of Computer and System Sciences*, 52(1): 43–52 (1996).

[52] V. R. Pratt. Every prime has a succinct certificate. *SIAM Journal on Computing*, 4: 214-220 (1975).

[53] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128-138 (1980).

[54] Ran Raz, Omer Reingold, Salil P. Vadhan. Extracting all the Randomness and Reducing the Error in Trevisan's Extractors. *Journal of Computer and System Sciences*, 65(1): 97–128 (2002).

[55] O. Reingold. Undirected st-connectivity in log-space. In *Proc. 37th ACM Symposium on the Theory of Computing*, 2005. To appear.

[56] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155(1): 157–187 (2001).

[57] M. Saks and S. Zhou. $BP_H SPACE(S) \subseteq DSPACE(S^{3/2})$. *Journal of Computer and System Sciences*, 58:376–403 (1999).

[58] M. Santha and U. V. Vazirani. Generating Quasi-random Sequences from Semi-random Sources. *Journal of Computer and System Sciences*, 33(1): 75–87 (1986).

[59] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2): 177–192 (1970).

[60] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4): 869-877 (1992).

[61] P. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proc. 35th Annual Symposium on Foundations of Computer Science*, 1994, 124–134.

[62] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6: 84–85 (1977).

[63] Róbert Szelepcsényi. The method of forcing for nondeterministic automata. *Bulletin of the European Association for Theoretical Computer Science*, 33:96–100 (1987).

[64] H. N. V. Temperley and M. E. Fisher. Dimer problem in statistical mechanics – an exact result. *Philosophical Magazine* 6: 1061– 1063 (1961).

[65] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4): 860–879 (2001).

[66] Salil P. Vadhan. The Complexity of Counting in Sparse, Regular, and Planar Graphs. SIAM J. Comput. 31(2): 398-427 (2001).

[67] Leslie G. Valiant. Expressiveness of Matchgates. *Theoretical Computer Science*, 281(1): 457-471 (2002). See also 299: 795 (2003).

[68] Leslie G. Valiant. Quantum circuits that can be simulated classically in polynomial time. *SIAM Journal of Computing*, 31(4): 1229-1254 (2002).

[69] Leslie G. Valiant. Holographic Algorithms (Extended Abstract). In *Proc. 45th IEEE Symposium on Foundations of Computer Science*, 2004, 306–315. A more detailed version appeared in Electronic Colloquium on Computational Complexity Report TR05-099.

[70] Leslie G. Valiant. Holographic circuits. In *Proc. 32nd International Colloquium on Automata, Languages and Programming*, 2005. To appear.

[71] Leslie G. Valiant. Completeness for parity problems. In *Proc. 11th International Computing and Combinatorics Conference*, 2005. To appear.

[72] U. V. Vazirani, V. V. Vazirani. Random Polynomial Time Is Equal to Slightly-random Polynomial Time In *Proc. 26th IEEE Symposium on Foundations of Computer Science*, 1985, 417–428.

[73] Andrew Chi-Chih Yao, Theory and Applications of Trapdoor Functions (Extended Abstract). In *Proc. 23rd IEEE Symposium on Foundations of Computer Science*, 1982, 80–91.