On the Intellectual Terrain Around NP*

Juris Hartmanis
Suresh Chari

TR 93-1402
December 1993

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

* This work is supported in part by NSF grant CCR-9123730.

On the Intellectual Terrain around NP[†]

Juris Hartmanis[1] and Suresh Chari[2]

[1] Cornell University and Max-Planck Institut für Informatik
[2] Cornell University

Abstract. In this paper we view $P \stackrel{?}{=} NP$ as the problem which symbolizes the attempt to understand what is and is not feasibly computable. The paper shortly reviews the history of the developments from Gödel's 1956 letter asking for the computational complexity of finding proofs of theorems, through computational complexity, the exploration of complete problems for NP and PSPACE, through the results of structural complexity to the recent insights about interactive proofs.

1 GÖDEL'S QUESTION

The development of recursive function theory, following Gödel's famous 1931 paper [Gö31] on the incompleteness of formal mathematical systems, clarified what is and is not effectively computable. We now have a deep understanding of effective procedures and their absolute limitations as well as a good picture of the structure and classification of effectively undecidable problems. It is not clear that the the concept of feasible computability can be defined as robustly and intuitively satisfying form as that of effective computability. At the same time it is very clear that with full awareness of the ever growing computing power we know that there are problems which remain and will remain, in their full generality, beyond the scope of our computing capabilities. One of the central tasks of theoretical computer science is to contribute to a deeper understanding of what makes problems hard to compute, classify problems by their computational complexity and yield a better understanding of what is and is not feasibly computable.

It is interesting to observe that the effort to understand what is and is not effectively computable was inspired by questions about the power of formal systems and questions about theorem proving. Not too surprisingly, the questions about the limits of feasible computability are also closely connected to questions about the computational complexity of theorem proving. It is far more surprising and interesting that it was again Gödel, who's work had necessitated the investigations of what is effectively computable, who asked the key question about feasible computability in terms of theorem proving. In a most interesting 1956 letter to his colleague at the Institute for Advanced Study, John von Neumann, Gödel asks for the computational complexity of finding proofs for theorems in formal systems(cf. [Har89]). Gödel is very precise, he specifies the Turing machine as a computational model and then asks for the function which bounds the number of steps needed to compute proofs

[†] This work is supported in part by NSF grant CCR–9123730

of length n. It does not take very long to realize that Gödel was, in modern terminology, asking von Neumann about the deterministic computational complexity of theorem proving and thus about the complexity of NP. In the same letter Gödel also asks about the computational complexity of problems such as primality testing and quite surprisingly, expresses the opinion that the problem of theorem proving may not be so hard. He mentions that it is not unusual that in combinatorial problems the complexity of the computation can be reduced from the N steps required in the brute force method to $\log N$ steps, or linear in the length of the input. He mentions that it would not be unreasonable to expect that theorem proving could be done in a linear or quadratic number of steps. Strange that the man who showed the unexpected limits of formal theorem proving did not seem to suspect that computational theorem proving and therefor NP may be at the limits of feasible computability. It is very unfortunate that von Neumann was already suffering from cancer and passed away the following year. No reply to this letter has been found and it seems that Gödel did not pursue this problem nor try to publicize it.

2 Complexity theory and complete problems

The full understanding of the importance of Gödel's question had to wait for the development of computational complexity which was initiated as an active computer science research area in the early 1960's. The basic hierarchy results were developed, the key concept of the complexity class, the set of all problems solvable(or languages acceptable) in a given resource bound, was defined and investigated[HS65, HLS65]. In a very general sense, the most important effect of this early work in complexity theory was the demonstration that there are strict quantitative laws that govern the behavior of information and computation. A turning point in this work came in 1971 with Cook's proof that SAT, the set of satisfiable Boolean formulas, was complete for NP, the class of nondeterministic polynomial time computations [Coo71](these results were independently discovered a year later by Levin [Lev73]). In other words, any problem that can be solved in polynomial time by guessing a polynomially long solution and then verifying that the correct solution had been guessed, could be reduced in polynomial time to SAT. In 1972 Karp [Kar72] showed that many combinatorial problems were in NP and could be reduce to SAT. On of the key problems in this set was the Clique decision problem: given a graph G and an integer k, determine if there is a clique of size k in this graph. Cook's and Karp's work virtually opened a flood gate of complete problems in NP and PSPACE. From all areas of computer science mathematics, operations research etc., came complete problems in all sizes and shapes all reducing to SAT and SAT reducing to them (see [GJ79] for a compendium).

Later, many other complexity classes were defined and found to have complete languages or problems. Among the better known ones are SPACE($\log n$), NSPACE($\log n$), P, NP, SPACE(n), NSPACE(n), the Polynomial Hierarchy(PH), PSPACE, NPSPACE(which is the same as PSPACE), EXPTIME, NEXPTIME, and EXPSPACE. The reassuring aspect of these classes is that they are very robust in their definition and that they appear naturally in many other settings which may or may not be directly connected to computing. For example, a variety of these complexity classes appear naturally as the classes definable by different logics, without any direct reference to computing(cf. [Imm89] for a survey). There is no doubt that complexity classes are a key concept and that they reveal a natural and important structure in the classification of computational problems and mathematical objects.

3 Structural Complexity Theory

The exploration of the structure of the complexity of computations has been an active area of research for more than a decade. This research explores the relations between various complexity classes and it investigates the internal structure of individual complexity classes. Figure 1 presents the key complexity classes below EXPSPACE, the class of all problems solvable with tape size bounded by an exponential in the length of the input. Besides the space and time bounded classes such as P, PSPACE etc. mentioned earlier the figure also shows the second level of the Polynomial Hierarchy in detail. $P^{SAT||[k]}$ is the class of languages recognizable by polynomial time oracle machines which can make up to k non–adaptive queries to a SAT oracle. The language of uniquely satisfiable formulas, USAT, is in Σ_2^p, the second level of the PH and also the complement of USAT is in Σ_2^p. In fact, USAT, is in the class $P^{SAT||[2]}$. $P^{SAT[\log n]}$ is the class of languages accepted by polynomial time oracle machines which make logarithmic number of queries to SAT and P^{SAT} is the class of languages recognizable with a polynomial number of queries to SAT. For several NP optimization problems where the optimum value is bounded by a polynomial in the input size, such as the Clique problem (the maximum clique size is the number of vertices), the optimum value can be computed with logarithmically many queries to SAT, by a binary search on the polynomial sized range of possible values. Optimization problems such as the Traveling Salesperson Problem can be solved with polynomially many queries to SAT.

Though we know a tremendous amount about the relations between these complexity classes, we still have no proof that they all are distinct: There is no proof that P is not equal to PSPACE! There is a very strong conviction that all the major complexity classes are indeed different, but in spite of all our efforts there is no nontrivial separation result. A major result would be any separation of: P and NP, NP and PSPACE, the various levels of the Polynomial Hierarchy, PSPSACE and EXPTIME, EXPTIME and NEXPTIME, and the last two from EXPSPACE. Less dramatic, but still very interesting would be the separation of the lower complexity classes, including SPACE($\log n$) and P or some even lower classes (defined by circuit models) not discussed here. It is clear that there are major problems facing complexity theory on which no substantial progress has been made since their definition, in some cases more than twenty years ago. The successes in recursive function theory have come from diagonalization arguments whose sharpness and sophistication has been elevated to a fine art. Unfortunately, though structural complexity theory has been strongly influenced by concepts from recursive function theory, diagonalization proof techniques have failed to dent the notorious separation problems. Since we can not diagonalize over these classes to construct a language just outside of the class, *i.e.* in a higher but not too high a class, we lack proof techniques to be able to deal with all possible ways of computing a problem in a given class to show that a certain problem from the class above can not be so computed. We badly need new concepts

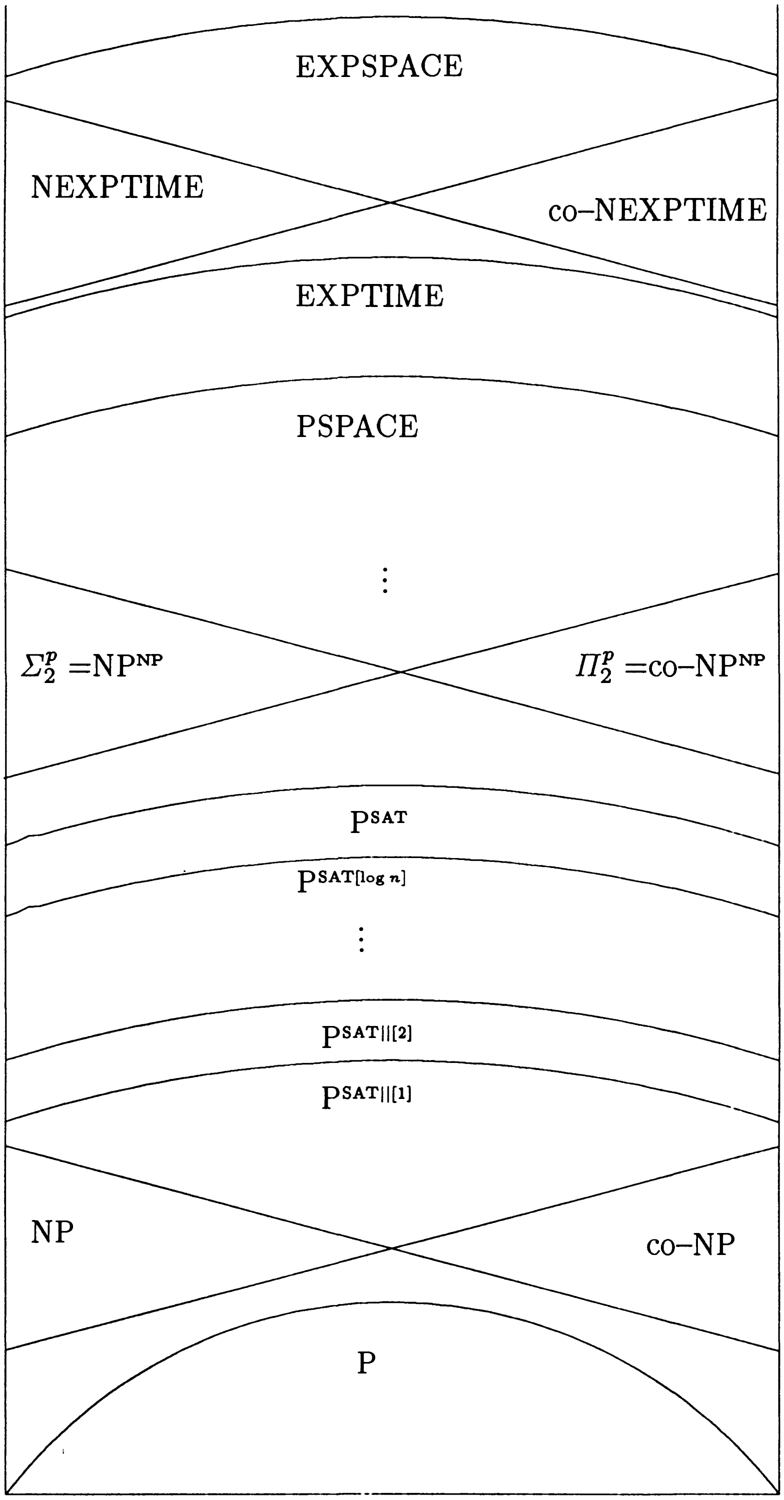

Fig. 1. Key complexity classes below EXPSPACE

and techniques to reason about all the possible computations in a complexity class to show their limitations.

Given the current situation, a working hypothesis in structural complexity theory is that P is different than NP and further more that the Polynomial Hierarchy is infinite. In particular, there have been many very interesting results which show that a given assumption implies that PH is finite, and thus giving in our view, a strong indication that the assumption is not true. We will illustrate some such results.

In 1977 Len Berman and the first author were led to conjecture on the strength of many special cases and with analogy to the corresponding situation in recursive function theory, that all NP complete sets are isomorphic under polynomial time computable isomorphisms [BH77]. This means that for any two NP complete problems there exists a bijection, reducing the problems to each other, and which is polynomial time computable in both directions, thus asserting that all NP complete problems are indeed very similar in structure. It is easily seen that this, the Berman–Hartmanis conjecture, implies that all NP complete sets must have roughly the same density. If we refer to a set as sparse if it has only polynomially many elements up to size n, then the conjecture implies that no sparse sets can be NP complete. Indeed, a former student of the first author, Steve Mahaney, proved the following very interesting result.

Theorem 1 (Mahaney [Mah82]). *If a sparse set is* NP *complete then* P=NP.

This leads us to accept, using our working hypothesis, that no sparse set can be complete for NP. Thus, there can not be a polynomially large dictionary which we can consult to solve an NP complete problem in polynomial time with its help. We describe the gist of the proof of this result with a new method due to Ogiwara and Watanabe [OW90], which has made a hard proof [Mah82] quite manageable. As you will see, the key ideas are deceptively simple, but please do not underestimate the originality and difficulty of the original proof and the very clever choice of the right NP complete set in the new proof.
Proof Outline: Instead of SAT, we will consider the set

$$A = \{(x, F) \mid F \text{ an } r\text{–variable formula, } |x| = r, \text{ and } (\exists y)\ x < y \text{ and } F(y) = 1\}.$$

It is easy to see that this set is NP complete; it is in NP and by fixing $x = 00....0$, it is SAT.

Let S be a sparse NP complete set, and let the density function of S, *i.e.* $|\{x|x \in S \text{ and } |x| \leq n\}|$, be bounded by n^k. Let f be the polynomial time computable function which reduces A to S with $|f(x)| \leq N(|x|)$. Think of the 2^r r-length binary strings arranged on a line in increasing order. Divide this line in $2(N(n))^k$ segments and compute the value of the reduction f of A to S at the $2(N(n))^k$ points at the beginning of each segment. Should all these values of f be different then we know that among the first $(N(n))^k + 1$ values there must be one that is not in S and thus shows that there is no y to the right of it which can satisfy F . Therefore we can remove the right half of the line since no solution exists there. We now repeat this process on the first half. Clearly, if our luck holds and, in all following computations the f values are distinct, in polynomially many rounds we will reach polynomially many values which must contain the satisfying assignment if there is one and we check all possibilities, solving the satisfiability problem in polynomial time.

Clearly, we can not expect that all the f values will be distinct all the time. At the same time, equal values also contain a lot of good information and can be used to eliminate from consideration the segments between any pair of values with same function value. To see this, observe that if for $x < x'$, we have $f(x) = f(x')$ not in S, then there is no solution to the right of x and therefore no solution between x and x'. If $f(x) = f(x')$ is in S, there are solutions to the right of x', and hence we can eliminate the segment between x and x' without losing all solutions. Again it is seen that if there are many equal values we toss out all the segments between pairs of equal values and the line is shortened. It is not too hard to see that combining both methods the total length of the remaining segments to be searched decreases by at least half each time and we can determine in polynomial time if a the formula F is satisfiable, which implies that P=NP.

To illustrate an assumption which forces the Polynomial Hierarchy to be finite we will consider deterministic polynomial time computations which can query a SAT-oracle, *i.e.* during the computation a polynomial number of questions can be asked about the outcome of NP computations. These considerations will also lead us to some recent interesting results about probabilistic computations which are intuitively very satisfying. To recall, $P^{\mathrm{SAT}||[k]}$ denotes the class of polynomial time computations with k parallel *i.e.* non-adaptive queries to the SAT oracle.

Theorem 2 (Kadin [Kad88]). *If for any k, $P^{\mathrm{SAT}||[k-1]} = P^{\mathrm{SAT}||[k]}$, then the Polynomial Hierarchy is finite.*

Not only can sparse sets not be NP complete if P is not equal to NP, but if the PH is infinite as we strongly expect, there is no way of even saving a single query to SAT out of millions of queries. On the other hand, our intuition suggests that as the number of queries to SAT increases the value of additional queries should decrease. Kadin's result shows that this is not the case, but there is another way of looking at this problem which indeed justifies our intuition of the decreasing value of queries as the numbers increases. To see this we consider randomized reductions:

Definition 3. A language $H \leq_m^{\mathrm{rp}}$ reduces to language K, with probability p if there exists a polynomial time function f such that

$$x \in H \Rightarrow \mathrm{Prob}[f(x,z) \in K] \geq p$$

$$x \notin H \Rightarrow \mathrm{Prob}[f(x,z) \notin K] = 1$$

when z is chosen at random from $q(n)$ long binary strings, where q is a polynomial.

The $\leq_m^{\mathrm{bpp}}$(two-sided error) reduction is defined similarly, with the condition being

$$\mathrm{Prob}[x \in H \text{ iff } f(x,z) \in K] > p.$$

The question now is, with what probability can languages in $\mathrm{P}^{\mathrm{SAT}||[k]}$ be reduced to $P^{\mathrm{SAT}||[k-1]}$. A very recent result by a former student of the first author, Pankaj Rohatgi, shows that there are very interesting threshold results about randomized reductions.

Theorem 4 (Rohatgi [Roh92]). *Every language in* $\mathrm{P}^{\mathrm{SAT}||[k]}$ *can be reduced to a language in* $\mathrm{P}^{\mathrm{SAT}||[k-1]}$ *with two-sided error reductions of probability* $1 - \frac{1}{k+1}$. *And this result is optimal: if there is a such a reduction possible with probability* $1 - \frac{1}{k+1} + \frac{1}{poly(n)}$ *then the* PH *is finite, where* n *is the size of the input to the reduction.*

This is a very interesting result: First it shows that with increasing k the value of additional queries to SAT decreases since the probability that it reduces to a problem solvable by one less query increases as k grows. At a million queries to SAT an additional query indeed has a small value. It also shows, surprisingly, that in randomized reductions there are very sharply defined threshold effects such that if it is possible to pass this threshold then the polynomial hierarchy collapses, just as in the deterministic case, according to Kadin's result. It is also interesting to note that the proof of this result uses the 'hard-easy' method used in Kadin's proof in a new setting with technically interesting methods. The results for the one-sided error case are similar but because of the stricter reductions the threshold probability is $1 - \frac{1}{\lceil k/2 \rceil}$. Again the violation of this threshold forces PH to be finite. The delightful part in the proofs is that it the reductions that achieve the desired probability bound are simple and it is surprising that the 'natural' method is also an optimal one. We urge you to read Rohatgi's PhD dissertation[Roh94].

4 Interactive Proofs

In 1985 two models of interactive randomized computation were introduced with very different motivations. In order to extend NP class slightly to capture some problems not known to be in NP, Babai defined the Arthur–Merlin games[Bab85](see also [BM88]). Motivated by possible applications to cryptographic protocols, Goldwasser, Micali and Rackoff defined the class of languages with interactive proofs[GMR89]. To motivate the introduction to this computing model, let us recall that NP is simply that class of problems for which, when a solution is suggested, it can be verified in polynomial time that the solution indeed solves the problem. We can think of this as a very powerful prover seeing the problem posed to the verifier and then simply sends him the solution, which than can be checked for validity by the verifier. Also, if no such solution exists, no prover, however powerful, can convince the verifier of the existence of a solution. Here the communication is one-way from the prover to the verifier. It is easily seen that adding a two way interaction in this model does not increase the computing power. To see this, we just have to observe that the verifier is a deterministic polynomial time machine and that the very powerful prover can compute the questions which the verifier will ask upon seeing the posed problem and in response to the answers from the old prover. Thus the prover after seeing the problem can send the whole exchange of questions and answers to convince the verifier that there is indeed a solution to the posed problem, and we are back to NP case. To have a unpredictable interaction between the prover and the verifier we need to add randomness to the arsenal of the verifier and give up the total certainty of verification of the solution as in the case of NP case. This leads us to the following definition of languages with interactive proofs.

Definition 5 ([GMR89]). A language L is in IP (*i.e.* it has an interactive proof) if there exists a verifier V, a randomized polynomial time machine such that

$$x \in L \Rightarrow \text{ There exists a prover } P^* \text{ such that } \mathrm{Prob}[(V, P^*) \text{ accept on } x] = 1$$

$$x \notin L \Rightarrow \text{ For all provers } P \ \mathrm{Prob}[(V, P) \text{ accept on } x] < \frac{1}{3}.$$

Note that repeated, independent executions of this protocol reduces the probability of failure exponentially fast. For some time after the introduction of these concepts it was not clear how powerful interactive proofs were. There is a very nice proof that the graph non-isomorphism problem, which is in co-NP and is not known to to be in NP, has an interactive proof [GMW86]. The protocol is indeed simple: For any pair of graphs (H, K), the Verifier selects randomly H or K, randomly permutes the graph, and asks the Prover to identify the graph as H or K. If H and K are not isomorphic a honest prover has no difficulty identifying the graph. If they are isomorphic, no Prover is capable of distinguishing between the randomly permuted versions of H and K, which are isomorphic. Thus, no prover can do any better than guessing the answer and hence cannot deceive the verifier with probability more than $\frac{1}{2}$. Again, by a repetition of this question-answer exchange, the probability that the verifier is fooled by a dishonest prover can be made exceedingly small.

It is also interesting to recall that there were oracle results[FS88] which gave relativized worlds in which co-NP was not contained in IP. This suggested, incorrectly, that IP was not very powerful or, according to the heuristic use of oracle results, indicating that it should be very hard to prove that co-NP is contained in IP in the real (unrelativized) world. The oracle hueristic implied that such a proof can not be obtained with “known methods”.

The determination of the full power of IP came very suddenly and it was a great surprise. In 1990, Lund, Fortnow, Karloff and Nisan [LFKN90] developed the techniques to show that the entire polynomial hierarchy is contained in IP and Shamir [Sha90] finished the characterization of the power of interaction and randomization by showing that IP=PSPACE. This was indeed a startling surprise with further unexpected consequences. The proof exploits in a very elegant way our better understanding of polynomials than Boolean formulas. The fact that polynomials can be ‘fingerprinted’ be evaluating them at a random point, was the key to the very ingenious proof. We know that if two polynomials, p and q of low degree evaluate to the same value with high probability at a randomly chosen point x, then, with high probability they are identical polynomials, since x is a root of the low degree polynomial $p(x) - q(x)$, and such polynomials have few roots unless identically zero. The key idea in the proof is to replace a given quantified Boolean formula, which constitute a PSPACE complete problem, QBF, by operations over a field of integers modulo a large prime. To convert a QBF into a polynomial, ‘or’ is replaced by $+$, ‘and’ by $\times$, ‘not’x by $(1-x)$, ‘for all’x by $\prod_{x=0,1}$, and ‘there exists’x by $\sum_{x=0,1}$. After that, a clever sequence of questions about the resulting polynomials and their evaluation (fingerprinting) at random points, yields the interactive protocol to test if the given QBF is satisfiable, thus showing that QBF$\in$IP, and since QBF is complete for PSPACE, PSPACE$\subseteq$IP. This startling result also gives a natural counterexample to refute the random oracle hypothesis. It is shown in [HCRR90b] that with probability 1, for a random oracle A, $\mathrm{IP}^A \neq \mathrm{PSPACE}^A$. The IP=PSPACE result also allows one to tie the ***width*** of a proof of a theorem in a formal system to the ease of convincing a verifier of its correctness with high probability[HCRR90a]. Informally, we can think of a proof of a theorem as being written on a two-dimensional page so that it can be verified easily, for *e.g.* by a finite automaton which can read one symbol on 2 adjacent lines of the proof. This can be shown to be a robust definition with the finite automaton replaced by several other models, all yielding the same class of proofs. It can then be shown that PSPACE is the class of languages with proofs with polynomially wide proofs, where the width of a 2-dimensional proof being the largest number of symbols on any one line. IP=PSPACE thus implies that width, as opposed to the length, of the proof determines how quickly one can give overwhelming evidence that a theorem is provable without showing its full proof.

The impact of the IP=PSPACE result is still being explored in complexity theory but some related results followed in rapid succession. Motivated again by their possible application in cryptographic protocols, Ben-Or, Goldwasser, Kilian and Wigderson [BGKW88] had earlier proposed, MIP, a multi-prover version of interactive proofs. Babai, Fortnow and Lund [BFL90] showed that this model, which has two provers who do not communicate with each other, increases the computational power substantially: MIP = NEXPTIME. Fortnow, Rompel and Sipser [FRS88] showed that the MIP model was equivalent to the following oracle model

Definition 6. A language L is in MIP if there exists a polynomial time probabilistic oracle machine V(the verifier) such that

$$x \in L \Rightarrow \text{ There exists oracle } O \text{ such that } \mathrm{Prob}[V^O(x) \text{ accepts }] = 1$$

$$x \in L \Rightarrow \text{ For all } O \ \mathrm{Prob}[V^O(x) \text{ accepts }] \leq \frac{1}{3}$$

Since the oracle answers can be thought of as an exponentially long 'proof' of membership, the result MIP=NEXPTIME, says that for languages in exponential time there are exponentially long proofs of membership which can be checked by inspection at a polynomial number of places. This characterization of MIP and the MIP=NEXPTIME were surprisingly used to show the hardness of approximating the clique problem[FGL+91] unless EXPTIME=NEXPTIME. This also initiated the attempt to 'scale down' the result on proofs whose correctness can be checked by inspection at very few places[BFLS91]. Two big breakthroughs followed: The results of [AS92] showed that indeed languages in NP had proofs which could be verified by checking at logarithmically many (in fact even smaller) places of the proof. Finally Arora *et al.* [ALM+92] showed that NP is the set of languages which have proofs which can be checked by verifiers which use logarithmically many random bits and which inspect the proof at only a constant number of positions! This surprising characterization has led to many results on the hardness of approximating many combinatorial problems[ALM+92, LY93, ABSS93], whose approximability has been open for several years.

References

[ABSS93] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes and systems of linear equations. In *Proceedings of the* 34^{th} *IEEE Symposium on Foundations of Computer Science*, pages 724–733, 1993.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. In *Proceedings of the* 33^{rd} *IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs. In *Proceedings of the* 33^{rd} *IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the* 17^{th} *Annual ACM Symposium on Theory of Computing*, pages 421–429, 1985.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two–prover interactive protocols. In *Proceedings of the* 31^{st} *IEEE Symposium on Foundations of Computer Science*, pages 16–25, 1990.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the* 23^{rd} *Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[BH77] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–322, 1977.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 34:254–276, 1988.

[BGKW88] M. Ben-or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi prover interactive proofs: How to remove intractability. In *Proceedings of the* 20^{th} *Annual ACM Symposium on Theory of Computing*, pages 113–131, 1988.

[Coo71] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the* 3^{rd} *Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[FGL+91] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Approximating clique is alomst NP-complete. In *Proceedings of the* 32^{nd} *IEEE Symposium on Foundations of Computer Science*, pages 2–12, 1991.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings of the* 3^{rd} *Structure in Complexity Theory Conference*, pages 156–161, 1988.

[FS88] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages? *Information Processing Letters*, 28(5):249–251, 1988.

[Gö31] K. Gödel. Über formal unentscheidbare Sätze der Principiua mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability:A guide to the theory of NP-Completeness.* Freeman, 1979.

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

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the* 27^{th} *IEEE Symposium on Foundations of Computer Science*, pages 174–187, 1986.

[Har89] J. Hartmanis. Gödel, von Neumann and the P=?NP problem. *Bulletin of the EATCS*, 38:101–107, June 1989.

[HCRR90a] J. Hartmanis, R. Chang, D. Ranjan, and P. Rohatgi. On IP=PSPACE and theorems with narrow proofs. *Bulletin of the EATCS*, 41:166–174, June 1990.

[HCRR90b] J. Hartmanis, R. Chang, D. Ranjan, and P. Rohatgi. Structural Complexity Theory: Recent Surprises. In *Proceedings of SWAT 90*, pages 1–12. Lecture Notes in Computer Science #447, 1990.

[HLS65] J. Hartmanis, P. Lewis, and R. Stearns. Hierarchies of memory limited computations. In *Proceedings of 6^{th} IEEE Symposium on Switching Circuit Theory and Logical Design*, pages 179–190, 1965.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Trans. AMS*, 117:285–306, 1965.

[Imm89] N. Immerman. Descriptive and computational complexity. In J. Hartmanis, editor, *Proceedings of Symposia in Applied Mathematics*, pages 75–91. AMS, 1989.

[Kad88] J. Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, 1988.

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Lev73] L. Levin. Universal'nyie perebornyie zadachi(universal search problems). *Problemy Peredachi Informatsii*, 9(3), 1973.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the* 31^{st} *IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990.

[LY93] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In *Proceedings of the 25^{th} Annual ACM Symposium on Theory of Computing*, pages 286–293, 1993.

[Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.

[OW90] M. Ogiwara and O. Watanabe. On poynomial time bounded truth-table reducibility of NP to sparse sets. In *Proceedings of the 22^{nd} Annual ACM Symposium on Theory of Computing*, pages 457–467, 1990.

[Roh92] P. Rohatgi. Saving queries with randomness. In *Proceedings of the 7^{th} Structure in Complexity Theory Conference*, pages 71–83, 1992.

[Roh94] P. Rohatgi. *On Properties of Random Reductions.* PhD thesis, Cornell University, 1994. Available as Computer Science Department technical report TR 93–1386.

[Sha90] A. Shamir. IP = PSPACE. In *Proceedings of the* 31^{st} *IEEE Symposium on Foundations of Computer Science*, pages 11–15, 1990.