SOME OBSERVATIONS ON THE PROBABILISTIC ALGORITHMS AND NP-HARD PROBLEMS

Ker-I KO
Department of Computer Science, University of Houston, Houston, TX 77004, U.S.A.

Received 18 February 1981; revised version received 30 June 1981

Probabilistic algorithms, polynomial time computation, nondeterminism, polynomial time hierarchy

1. Introduction

Recently polynomial time probabilistic algorithms with small error probability for the primality problem have been discovered by Rabin [8] and Solovay and Strassen [11]. Their algorithms share a special property that they always identify prime numbers correctly. We call this kind of algorithm Rabin-type algorithms. Since the primality problem is in NP $\cap$ co-NP but not known to be in P [6,7], it is suspected that probabilistic algorithms are more powerful than deterministic algorithms at the polynomial time level. Another interesting question is whether probabilistic algorithms are more powerful than nondeterministic algorithms. In particular, is there any polynomial time probabilistic algorithm with small error probability for some NP-hard problem? Is there any Rabin-type algorithm for some NP-complete problem? Gill gave partial answers to these questions [3]. He showed that every problem in NP can be computed by some polynomial time probabilistic algorithm with error probability less than $\frac{1}{2}$. However, it is not known whether such probabilistic algorithms can be improved to have error probabilities bounded by a constant less than $\frac{1}{2}$. Gill also mentioned that little is known about the relation between the class of polynomial time probabilistically computable sets and the polynomial time hierarchy of Meyer and Stockmeyer [12]. Simon mentioned that a probabilistic hierarchy can be constructed and conjectured that two hierarchies are independent [10].

In this note we further study the relationship between complexity classes defined by nondeterministic and probabilistic computations. We show that the question whether a problem in the polynomial time hierarchy has a probabilistic algorithm with small error bound is reduced to the question whether an NP-complete problem has such a probabilistic algorithm. Furthermore, both of the above questions are reduced to the question whether an NP-complete problem has a Rabin-type algorithm or not. Recent studies on the relativized version and the circuit complexity of the last question showed that this question is probably as difficult as the well-known P = ?NP question [9,2,5]. Therefore, we conclude that there are no probabilistic algorithms with small error bounds for NP-hard problems unless there is a big theoretical breakthrough. We also propose a different probabilistic hierarchy using Rabin-type algorithms as a basis. We show that the relationship between this hierarchy and the polynomial time hierarchy can be reduced to the same question.

2. Preliminaries

Gill [3] used the probabilistic Turing machine (PTM) as a model of probabilistic computation. We shall follow his definitions and terminology.

A PTM is a Turing machine equipped with a random state. For each random state, the control unit specifies two possible next states. When the machine enters a random state, it tosses an unbiased coin (or, generates a random bit 0 or 1) to decide with which of the two next states to continue. In general each PTM M computes a random function. Here, we restrict ourselves to language-recognition problems. We say that a set A is recognized by a PTM M if $\Pr\{M(x) = \chi_A(x)\} > \frac{1}{2}$ for all x, where χ_A is the characteristic function of A. The time complexity T_M of a PTM M can be similarly defined. Let A be the set recognized by M. Then $T_M(x)$ is the least positive integer n such that $P_r\{M(x) = \chi_A(x) \text{ in n steps}\} > \frac{1}{2}$. We say that a PTM M is polynomial time bounded *if* there is a polynomial p such that $T_M(x) \leq p(|x|)$ for all x ($|x|$ is the length of the string x). The error bound of a PTM M is naturally defined to be $\sup_x \Pr\{M(x) \neq \chi_A(x)\}$.

Now we define some interesting complexity classes based on probabilistic computation.

PP is the class of languages recognized by polynomial time PTM's.

BPP is the class of languages recognized by polynomial time PTM's with an error bound $c < \frac{1}{2}$.

R is the class of languages recognized by polynomial time PTM's which have zero probability of error for inputs in the complements of the languages. (Gill called this class VPP. Our notation comes from [1].)

Gill has discovered the following relations among these classes and the classes P (polynomial time recognizable languages), NP (polynomial time nondeterministically recognizable languages) and PSPACE (polynomial space recognizable languages):

$$P \subseteq R \subseteq BPP \subseteq PP \subseteq PSPACE$$

and

$$R \subseteq NP \subseteq PP\,.$$

None of the above inclusions are known to be proper. He also mentioned that it does not appear either $NP \subseteq BPP$ or $BPP \subseteq NP$.

In the following we define the polynomial time hierarchy [12]. If A is a set of strings, let P^A (NP^A) be the class of languages recognized by polynomial time oracle deterministic (nondeterministic) Turing machines with the oracle A. If $\mathcal{C}$ is a collection of sets, then $P^{\mathcal{C}} = \bigcup_{A\in\mathcal{C}} P^A$ and $NP^{\mathcal{C}} = \bigcup_{A\in\mathcal{C}} NP^A$. The polynomial time hierarchy is defined recursively:

$$\Sigma_0^p = \Pi_0^p = \Delta_0^p = P\,,$$

and for $k \geq 0$,

$$\Sigma_{k+1}^p = NP^{\Sigma_k^p}\,, \qquad \Pi_{k+1}^p = \{A\colon \bar{A} \in \Sigma_{k+1}^p\}\,,$$

$$\Delta_{k+1}^p = P^{\Sigma_k^p}\,.$$

Let $PH = \bigcup_{k=0}^{\infty} \Sigma_k^p$. It is known that $PH \subseteq PSPACE$ [12].

3. Main results

The main question we are interested in is whether $NP \subseteq BPP$ or not. We will show that it is equivalent to NP = R. First we observe that repeated simulations of a probabilistic algorithm with small error probability can help to eliminate the requirement of an oracle.

Theorem 1. $BPP^{BPP} = BPP$.

Proof. It suffices to show that $BPP^{BPP} \subseteq BPP$.

First note that we may assume that a set in BPP (or in BPP^{BPP}) can be recognized by some PTM (or some oracle-PTM) with any small error probability since we can simply repeat its recognition a sufficiently large number of times to reduce its error probability.

Let A be a set in BPP^{BPP}. Then there is an oracle PTM M_1 which recognizes A with the help of an oracle B in BPP. There is also a PTM M_2 recognizing B. Assume that the time complexity of M_1 and M_2 are bounded by two polynomials p_1 and p_2, respectively. We further assume that the *maximum* run time of M_2 is also bounded by p_2. (This can be done by attaching a 'clock' machine to the original probabilistic machine for B to keep track of the number of moves made by the machine.) Also assume that the error bounds for both M_1 and M_2 are less than $\frac{1}{16}$. In the following we describe a probabilistic algorithm M for A which simulates M_1 and M_2.

(Algorithm M.) For a given input x whose length is n, simulate M_1 on x. During the simulation, we may encounter some queries like 'Is y in B?'. For each query, we simulate M_2 on y for $(2 \cdot p_1(n) + 1)$ times and take the majority answer.

We need to check the error bound and the time bound of the algorithm M.

(Error Bound.) First observe that for any x, if (1) all the simulations of '$y \in B$?' queries give correct answers and (2) the simulation of M_1 on x is correct then $M(x) = \chi_A(x)$. Since the simulations in (1) and

(2) are mutually independent, we have

$$\Pr\{M(x) = \chi_A(x)\} \geqslant$$

$$\geqslant \left[\prod_{\substack{y \in \text{Queries made in} \\ \text{computation of } M_1^B(x)}} \Pr\{M_2(y) = \chi_B(y)\}\right]$$

$$\cdot \Pr\{M_1^B(x) = \chi_A(x)\} .$$

To see that this value is greater than $\frac{1}{2}$, we first observe that:

Lemma 1. For every y,

$\Pr\{$the majority result of $(2 \cdot p_1(n) + 1)$ many$\}$
$< 2^{-p_1(n)}$,
$\Pr\{$simulations of $m_2(y) \neq \chi_B y\} < 2^{-p_1(n)}$.

Proof. See Appendix.

Also, the computation of M_1^B on x halts in $p_1(n)$ steps if $M_1^B(x) = \chi_A(x)$. Therefore, in such computations M_1 can make at most $p_1(n)$ many queries. Therefore,

$$\Pr\{M(x) = \chi_A(x)\}$$

$$\geqslant (1 - 2^{-p_1(n)})^{p_1(n)} * \tfrac{15}{16} .$$

A simple mathematical induction shows that $(1 - 2^{-k})^k \geqslant \frac{9}{16}$ for all $k \geqslant 2$. Thus we have established an error bound $\frac{135}{256}$ for M.

(Time Bound.) For the input x on which M_1^B outputs exactly $\chi_A(x)$ and all simulations of $M_2(y)$ outputs correctly $\chi_B(y)$, M_1^B may ask at most $p_1(n)$ many queries '$y \in B$?', each y of length $\leqslant p_1(n)$. Since the maximum run time of M_2 on y is bounded by $p_2(|y|)$, the total run time is bounded by $p_1(n) + p_1(n) \cdot (2 \cdot p_1(1) + 1) \cdot p_2(p_1(n))$, a polynomial in n.

The following stronger version of Theorem 1 will be used later. Its proof is essentially the same as that of Theorem 1. Let FBPP denote the class of *functions* computable probabilistically in polynomial time with error probability bounded by a constant less than $\frac{1}{2}$.

Corollary 1. $\text{FBPP}^{\text{BPP}} = \text{FBPP}$.

From the definition of the polynomial time hierarchy, we have immediately the following corollary.

Corollary 2. If NP $\subseteq$ BPP then PH $\subseteq$ BPP.

A consequence of the result that PH $\subseteq$ BPP is that all NP-hard optimization problems have some probabilistic algorithms with small error bounds because they are in the class Σ_2^P. For instance, let us consider the Minimum Graph Coloring problem.

Let G = (V, E) be an undirected graph with $|V| = n$ vertices. A coloring c of the graph G is a function c: $V \to \{1, 2, ..., n\}$. We can also represent c as an n-tuple element in $T_n = \{1, 2, ..., n\}^n$ i.e. $c = (c(v_1), c(v_2), ..., c(v_n)) = (c_1, ..., c_n)$. We say that a coloring c of G separates vertices of G and write S(c, G) if $c(v_i) \neq c(v_j)$ for all $\{v_i, v_j\} \in E$. The number of colors used by c is the cardinality of the set $\{c(v_1), ..., c(v_n)\}$ and is represented by K(c). The Minimal Graph Coloring problem (Min-GC) is to find, for a given graph G, a coloring c of G such that S(c, G), and that K(c) is the minimum. As a language recognition problem, let

$$\text{GC} = \{(G, h): (\exists c \in T_n)[S(c, G) \text{ and } K(c) \leqslant h]\} .$$

It is well-known that GC is NP-complete [4].

Now, assume that Min-GC has a probabilistic algorithm A. Then it is reasonable to assume that, for a given input graph G, A always returns a coloring c = A(G) which separates the vertices of G and with more than $\frac{1}{2}$ the probability K(c) is the minimum. Based on this assumption, it is easy to construct a Rabin-type algorithm for recognizing GC:

(Algorithm for GC.) For a given G and a positive integer h, simulate A on G. If A(G) uses more than h colors then output 0; otherwise output 1.

Note that if $(G, h) \notin$ GC, then A(G) will always use more than h colors and if $(G, h) \in$ GC then $\Pr\{K(A(G)) \leqslant h\} > \frac{1}{2}$. Thus this is a Rabin-type algorithm.

Formulating the above observation at the formal language level we obtain the following lemma.

Lemma 2. If Min-GC $\in$ BPP then GC $\in$ R and hence NP = R.

Proof. We defined BPP as a collection of languages. So we need to give a language-recognition definition for Min-GC and modify the above simple argument for the language Min-GC.

In the following, the ordering $<$ on $T_n = \{1, 2, ..., n\}^n$ denotes the lexicographic ordering on T_n.

We define

$$f(G) = \max\{d \in T_n : S(d, G) \; \& \; (\forall e \in T_n) \times [S(e, G) \to K(d) \leqslant K(e)]\}$$

and

$$\text{Min-GC} = \{(G, c) \colon c \in T_n \; \& \; c \leqslant f(G)\} .$$

The function f can be calculated by binary searching over Min-GC. By Corollary 1, if Min-GC $\in$ BPP then there is a probabilistic algorithm M_f computing f with small error. Furthermore, M_f can be used to solve the questions '$(G, h) \in GC$?':

(Algorithm.) For an input (G, h), find $M_f(G) = d$. If $S(d, G)$ and $K(d) \leqslant h$ then output 1; otherwise, output 0.

(Error Probability.) If $(G, h) \notin GC$, then there is no d such that $S(d, G)$ and $K(d) \leqslant h$. Therefore, 0 will always be output. If $(G, h) \in GC$, then $\Pr\{S(M_f(G), G) \text{ and } K(M_f(G)) \leqslant h\} \geqslant \Pr\{M_f(G) = f(G)\} > \frac{1}{2}$.

Therefore, min-GC $\in$ BPP $\Rightarrow$ GC $\in$ R.

Finally observe that sets many-one reducible in polynomial time to a set in R are also in R. So, the fact that GC is NP-complete implies that Min-GC $\in$ BPP $\Rightarrow$ NP = R.

The above simple argument can be applied to many other NP-hard optimization problems whose corresponding yes/no problems are NP-complete. They include most of Karp's original list of NP-complete problems such as the Traveling Salesperson Problem, the Minimum Set Covering Problem and the Maximal Clique Problem [4].

Also note that Min-GC (the language defined in the proof of Theorem 2) is in Σ_2^P because Min-GC can be expressed as an $\exists\forall$-form predicate. An immediate consequence of Lemma 2 is then the following.

Corollary 3. If NP $\neq$ R then $\Sigma_2^P \not\subseteq$ BPP.

Combining Corollaries 2 and 3 we have

Theorem 2. NP $\subseteq$ BPP $\Rightarrow$ NP = R.

Therefore, the question whether R = NP is a fundamental question in the relation between probabilistic and nondeterministic algorithms. Unfortunately, the following results suggested that there is no easy solution to this question.

(1) (Rackoff [9].) There exist recursive sets A and B such that (i) $P^A = R^A \neq NP^A$, and (ii) $P^B \neq R^B = NP^B$.

(2) (Karp and Lipton [5].) NP $\neq$ R unless PH = Δ_2^P.

Karp and Lipton's result seems to support the conjecture that NP $\neq$ R for which Bennett and Gill [2] gave more evidence.

Simon [10] mentioned using alternative existential and threshold quantifiers to construct a probabilistic hierarchy. However, it is not clear how we may relate it to a hierarchy defined by oracle-computation. We feel that a more interesting hierarchy is a subhierarchy of PH based on the set R because we believe that R $\neq$ co-R, and the structure of this hierarchy is then similar to that of the polynomial time hierarchy.

Definition. The R-hierarchy consists of the following classes:

$$\Sigma_0^r = \Pi_0^r = \Delta_0^r = P ,$$

and for $k \geqslant 0$,

$$\Sigma_{k+1}^r = R^{\Sigma_k^r} , \qquad \Pi_{k+1}^r = \text{co-}\Sigma_k^r \quad \text{and} \quad \Delta_{k+1}^r = P^{\Sigma_k^r} .$$

Also, let

$$RH = \bigcup_{k=0}^{\infty} \Sigma_k^r .$$

From the relation R $\subseteq$ NP, we have immediately RH $\subseteq$ PH. Also, R $\subseteq$ BPP implies RH $\subseteq$ BPP (by Theorem 1). Therefore, we have:

Corollary 4. NP $\not\subseteq$ RH unless R = NP.

Thus, the relationship between RH and PH is also dependent on the question R = ?NP.

4. Appendix

Lemma 1. If the probability of a single event E is $\frac{15}{16}$, then in a sequence of $(2n + 1)$ repeated independent trials the probability of having at least $n + 1$ occurrences of E is $> 1 - 2^{-n}$.

Proof. Let q_i be the probability of exactly i occurrences of E in (2n + 1) repeated trials. Then

$$p = \Pr\{\text{at least } (n+1) \text{ occurrences of E}\}$$
$$= \sum_{i=n+1}^{2n+1} q_i = 1 - \sum_{i=0}^{n} q_i .$$

It is easy to see that $q_0 \leqslant q_1 \leqslant \cdots \leqslant q_n$ since $\Pr\{E\} = \frac{15}{16}$. Also,

$$q_n = \binom{2n+1}{n}\left(\frac{1}{16}\right)^{n+1}\left(\frac{15}{16}\right)^n \leqslant \binom{2n+1}{n}\left(\frac{1}{16}\right)^{n+1} .$$

So, we have

$$p \geqslant 1 - (n+1)\cdot q_n \geqslant 1 - \frac{(2n+1)!}{(n!)^2} \cdot 2^{-4(n+1)} .$$

By the Stirling's approximation to n!,

$$\sqrt{2\pi n}(n/e)^n < n! < \sqrt{2\pi n}(n/e)^n \cdot (\tfrac{3}{2})$$

we have

$$\frac{(2n+1)!}{(n!)^2} \leqslant \frac{(2n+1)\cdot 2^{2n}\cdot(3/2)}{\sqrt{\pi n}}$$
$$\leqslant (2n+1)\cdot 2^{2n} \leqslant 2^{3n+1} .$$

So,

$$p \geqslant 1 - 2^{3n+1}\cdot 2^{-4n-4} \geqslant 1 - 2^{-n} .$$

References

[1] L. Adleman and K. Manders, Reducibility, randomness, and intractability, Ninth Annual ACM Symposium on Theory of Computing (1977) pp. 151–153.

[2] C.H. Bennett and J. Gill, Relative to a random oracle A, $P^A \neq NP^A \neq$ co-NP^A with probability 1, SIAM J. Comput. 10 (1981) 96–113.

[3] J. Gill, Computational complexity of probabilistic Turing machines, SIAM J. Comput. 6 (4) (1977) 675–695.

[4] R.M. Karp, Reducibilities among combinatorial problems, in: R.E. Miller and J.W. Thatcher, Eds., Complexity of Computer Computations (Plenum Press, New York, 1972) pp. 85–104.

[5] R. Karp and R. Lipton, Some connections between nonuniform and uniform complexity classes, Twelfth ACM Symposium on Theory of Computing (1980) pp. 302–309.

[6] G.L. Miller, Riemann's hypothesis and tests for primality, J. Comput. System Sci. 13 (1976) 300–317.

[7] V. Pratt, Every prime has a succinct certificate, SIAM J. Comput. 4 (1975) 214–220.

[8] M.O. Rabin, Probabilistic algorithms, in: J.F. Traub, Ed., Algorithms and Complexity (Academic Press, New York, 1976) pp. 21–39.

[9] C. Rackoff, Relativized questions involving probabilistic algorithms, Proc. Tenth ACM Symposium on Theory of Computing (1978) pp. 338–342.

[10] J. Simon, On some central problems in computational complexity, TR 75-224, Cornell University, Ithaca, NY (1975).

[11] R. Solovay and V. Strassen, A fast Monte-Carlo test for primality, SIAM J. Comput. (1977) 84–85.

[12] L.J. Stockmeyer, The polynomial time hierarchy, Theoret. Comput. Sci. 3 (1) (1977) 1–22.