数理解析研究所講究録
第 695 巻 1989 年 198-204

確率的多項式時間アルゴリズムの能力について

戸田誠之助（電気通信大学・情報工学科）

Abstract. In this article, we investigate the computational power of probabilistic Turing machines. Let PP(A) denote the class of sets accepted by polynomial time-bounded probabilistic Turing machines with two-sided unbounded error probability. We show that PP(FewP) = PP(SPARSE∩NP) = PP(BPP) = PP, where FewP denotes the class of sets accepted by NP-machines with the property that for all inputs x, the number of accepting paths is bounded above by a polynomial in $|x|$, SPARSE denotes the class of sparse sets, and BPP denotes the class of sets accepted by polynomial time-bounded probabilistic Turing machines with two-sided bounded error probability. Furthermore, we observe that all equivalences above can be relativized with all oracles (i.e., PP(FewP(A)) = PP(SPARSE∩NP(A)) = PP(BPP(A)) = PP(A) for all oracles A). Our motivation of this work is to find the precise relationship between PP and PH (the polynomial-time hierarchy). In particular, whether PH is included in PP is the most important question. An approach to this question is to ask whether PP(NP) $\subseteq$ PP. If one could settle this question by using a relativizable technique, then he will observe PH $\subseteq$ PP. Thus our work can be viewed as a case study of the original question which has remained open.

1 Preliminaries.

We assume that the reader is familiar with the basic concepts of computational complexity theory. Let Σ be a finite alphabet. Our sets in this paper are all over $\{0,1\}$ unless otherwise specified. For a string $w \in \Sigma^*$, $|w|$ denotes the length of w. For a set $L \subseteq \Sigma^*$, $\overline{L}$ denotes the complement of L. For a class $\mathbf{C}$ of sets, co–$\mathbf{C}$ denotes the class of sets whose complement is in $\mathbf{C}$. Let Σ^n (resp., $\Sigma^{\leq n}$ and $\Sigma^{<n}$) denote the set of strings with length n (resp., at most n and less than n). For a finite set $X \subseteq \Sigma^*$, $\|X\|$ denotes the number of strings in X. Let $\mathbf{N}$ denote the set of natural numbers. All natural numbers are encoded in binary unless otherwise specified. $\langle \cdot, \cdot \rangle$ denotes a pairing function on Σ^* which is polynomial-time computable and whose left and right inverses are also polynomial-time computable. A k-tuple function on Σ^* is defined by a usual manner.

We also assume that the reader is familiar with standard complexity classes such as P, NP, and the polynomial-time hierarchy. We abbreviate by an *oracle P-machine* (resp.,

an *oracle NP-machine*) a polynomial time-bounded deterministic (resp., nondeterministic) oracle machine. An oracle NP-machine is called an *oracle FewP-machine* iff for each oracle set A and each input x, it has at most a polynomial number of accepting paths. FewP(A) denotes the class of sets accepted by oracle FewP-machines with oracle set A. The class FewP is defined as the class FewP($\emptyset$). PP(A) is the class of sets accepted by polynomial time-bounded probabilistic oracle Turing machines with two-sided unbounded error probability which consult to an oracle set A. In other word, a set L is in PP(A) if there exists an oracle NP-machine M such that for all x, $x \in$ L iff more than half of computation paths of M(A) on input x is accepting. The class PP is defined as the class PP($\emptyset$).

#P(A) is the class of functions which gives the number of accepting paths of oracle NP-machines with oracle set A. PF is the class of polynomial-time computable functions. All functions in this paper are ones from strings to natural numbers unless otherwise specified. It is well known that PP and #P are closely related to each other. Some relationships are mentioned in the next section and are used to prove our main results.

We assume that all oracle machines M satisfy the following conditions.

(1) Its transition function has exactly two possible transitions from each configuration.

(2) There exists a polynomial p such that for all oracles A and all input x, each computation path of M(A) on x from the initial configuration to a halting configuration is of length $p(|x|)$ exactly.

(3) All computation paths of it is encoded into a string of $\{0,1\}^*$ by a usual manner, where a computation path may contain possible answers from a given oracle and the oracle answer "yes" (resp., "no") is encoded by 0 (resp., 1).

These assumptions are technical ones. Obviously, we do not loose the generality under these assumptions.

A set L is said to be *PP-low* iff PP(L) = PP. A class **C** of sets is said to be *PP-low* iff all sets in the class are PP-low. It is trivial that for any class **C**, it is PP-low iff PP(**C**) = PP. Thus we will use the phrasing "a class **C** is PP-low" instead of describing the equality PP(**C**) = PP. A notion of *low sets* was first introduced by Shöning [8] within the polynomial-time hierarchy. That notion has been used to clasify decision problems and to show some structural differencies of NP-complete sets from sets with special properties [8,5,7].

2 Technical lemmas.

Before proving main results of this paper, we prepare some technical lemmas. Although we omit the proofs of these lemmas, almost all of these are not difficult.

Lemma 2.1 *For any sets L and A, the following statements are equivalent.*

(1) *L is in PP(A).*

(2) *There exist two functions $f, g \in \#P(A)$ such that $L = \{ x : f(x) \geq g(x) \}$.*

Lemma 2.2 *Let g, f be functions in $\#P(A)$ and let g give the number of accepting paths of an oracle NP-machine M with oracle set A. Let t be a polynomial bounding the run time of M. Then, h_1, h_2 and h_3 defined below are in $\#P$.*

$$h_1(x) = f(x) + g(x),\ h_2(x) = f(x)g(x),\ h_3(x) = 2^{t(|x|)} - f(x).$$

Lemma 2.3 *Let $g : \Sigma^* \times \mathbf{N} \to \mathbf{N}$ be a function in $\#P$, let e, f be a function in PF, and let A be a set of natural numbers recognizable in polynomial time. Then, h defined below is in $\#P$.*

$$h(x) = \sum_{e(x) \leq i \leq f(x), i \in A} g(x, i).$$

Lemma 2.4 *For each set L, $L \in PP(FewP)$ iff there exist a set A in FewP, a polynomial p, and a function f in PF such that for each x,*

$$x \in L \text{ iff } \|\{w \in \Sigma^{\leq p(|x|)} : \langle x, w \rangle \in A\}\| \geq f(x).$$

Lemma 2.5 *For each $k \geq 1$, $-\Sigma_{1 \leq i \leq k}(-1)^i \binom{k}{i} = 1$.*

3 PP-low classes: FewP, Sparse sets in NP, and BPP.

In this section, we prove the following theorem.

Theorem 3.1 *FewP is PP-low. Namely, PP(FewP) = PP.*

The inclusion PP $\subseteq$ PP(FewP) is obvious. The proof of the converse inclusion is quite involved and is done by a sequence of lemmas.

Let L be a set in PP(FewP). Let A, p and f be the same ones as in Lemma 2.4. Then, for each x,

$$x \in L \text{ iff } \|\{w \in \Sigma^{\leq p(|x|)} : \langle x, w\rangle \in A\}\| \geq f(x).$$

Let M_a be a FewP-machine which accepts A and let q be a polynomial bounding the number of accepting paths of M_a. Let r be a polynomial satisfying that $r(n) \geq \max\{$ $|\langle x, w\rangle| : |x| = n$ and $|w| \leq p(n)$ $\}$ for each $n \geq 0$. We note that for each input $\langle x, w\rangle$ such that $|w| \leq p(|x|)$, the number of acceptng paths of M_a is bounded above by $q(r(|x|))$. We define a set B and a function g by

$$\begin{aligned} B = \{\ \langle x, i, w, Q\rangle : {} & 1 \leq i \leq q(r(|x|)),\ |w| \leq p(|x|), \\ & Q \text{ is a set of accepting paths of } M_a \text{ on input } \langle x, w\rangle, \text{ and } \|Q\| = i\ \}, \\ g(x, i) = {} & \|\{\langle w, Q\rangle : \langle x, i, w, Q\rangle \in B\}\|. \end{aligned}$$

In the above definitions, a finite set Q is encoded as a lexicographically ordered list of elements in the set.

It is not hard to see that B is in P and hence g is in #P. Let t be a polynomial such that for each $\langle x, i\rangle$ satisfying $0 \leq i \leq q(r(|x|))$,

$$t(|x|) \geq max\{|\langle w, Q\rangle| : \langle x, i, w, Q\rangle \in B\}.$$

Then we define two functions g', h as follows:

$$g'(x, i) = \begin{cases} 2^{t(|x|)} - g(x, i) & \text{if } 0 \leq i \leq q(r(|x|)) \\ 0 & \text{otherwise} \end{cases}$$

$$h(x) = \sum_{1 \leq i \leq q(r(|x|)), i \text{ is odd}} g(x, i) + \sum_{1 \leq i \leq q(r(|x|)), i \text{ is even}} g'(x, i).$$

It is easy to see from Lemma 2.2 and Lemma 2.3 that g' and h are in #P. Now, we would like to prove the following lemma.

Lemma 3.2 *For each x,*

$$x \text{ is in } L \text{ iff } h(x) \geq f(x) + 2^{t(|x|)} \cdot \lfloor q(r(|x|))/2 \rfloor \cdot (\lfloor q(r(|x|))/2 \rfloor + 1).$$

From this lemma and Lemma 2.1, we conclude L is in PP. To prove this lemma, we need more definitions.

For each x and each $1 \leq k \leq q(r(|x|))$, we define a set $C(x, k)$ by

$$\begin{aligned} C(x, k) = \{w \in \Sigma^{\leq p(|x|)} : {} & \text{the number of accepting paths of } M_a \text{ on input} \\ & \langle x, w\rangle \text{ is exactly } k\ \}. \end{aligned}$$

Lemma 3.3 *For each x,*

$$x \in L \text{ iff } \sum_{1 \le k \le q(r(|x|))} \|C(x,k)\| \ge f(x).$$

Proof. It is easy to see that $C(x,k) \neq C(x,k')$ for each x and each $1 \le k \neq k' \le q(r(|x|))$. Furthermore, for each x, $\bigcup_{1 \le k \le q(r(|x|))} C(x,k) = \{ w : |w| \le p(|x|) \text{ and } \langle x,w \rangle \in A \}$. Hence, for each x,

$x \in L$
iff $\|\{ w : |w| \le p(|x|) \text{ and } \langle x,w \rangle \in A \}\| \ge f(x)$
iff $\Sigma_{1 \le k \le q(r(|x|))} \|C(x,k)\| \ge f(x)$. □

Lemma 3.4 *For each $\langle x,i \rangle$ satisfying $1 \le i \le q(r(|x|))$,*

$$g(x,i) = \sum_{i \le k \le q(r(|x|))} \binom{k}{i} \|C(x,k)\|.$$

Proof. From the definition of g,

$$\begin{aligned} g(x,i) &= \Sigma_{i \le k \le q(r(|x|))} \Sigma_{w \in C(x,k)} \|\{S \subseteq ACC(M_a, \langle x,w \rangle) : \|S\| = i\}\| \\ &= \Sigma_{i \le k \le q(r(|x|))} \binom{k}{i} \|C(x,k)\|, \end{aligned}$$

where $ACC(M_a, \langle x,w \rangle)$ denotes the set of accepting paths of M_a on input $\langle x,w \rangle$. □

Lemma 3.5 *For each x,*

$$\begin{aligned} h(x) = \; & \Sigma_{1 \le k \le q(r(|x|))} \{ -\|C(x,k)\| \cdot \Sigma_{1 \le i \le k} (-1)^i \binom{k}{i} \} \\ & + 2^{t(|x|)} \cdot \lfloor q(r(|x|))/2 \rfloor \cdot (\lfloor q(r(|x|)) \rfloor + 1). \end{aligned}$$

Proof. From the definition of h and Lemma 3.4,

$$\begin{aligned} h(x) = \; & \{ \Sigma_{1 \le i \le q(r(|x|)), i \text{ is odd}} g(x,i) - \Sigma_{1 \le i \le q(r(|x|)), i \text{ is even}} g(x,i) \\ & + \Sigma_{1 \le i \le q(r(|x|)), i \text{ is even}} 2^{t(|x|)}. \\ = \; & \Sigma_{1 \le k \le q(r(|x|))} \{ -\|C(x,k)\| \cdot \Sigma_{1 \le i \le k} (-1)^i \binom{k}{i} \} \\ & + 2^{t(|x|)} \cdot \lfloor q(r(|x|))/2 \rfloor \cdot (\lfloor q(r(|x|)) \rfloor + 1). \end{aligned}$$

Hence we have this lemma. □

Proof of Lemma 3.2. From Lemma 3.5 and Lemma 2.5, for each x,

$$h(x) = \sum_{1 \le k \le q(r(|x|))} \|C(x,k)\| + 2^{t(|x|)} \cdot \lfloor q(r(|x|))/2 \rfloor \cdot (\lfloor q(r(|x|))/2 \rfloor + 1).$$

Hence we have this lemma from Lemma 3.3. □

We also have other PP-low classes. A set S is *sparse* iff there exists a polynomial p such that for all $n \geq 0$, $\|\{ x \in S : |x| = n \}\| \leq p(n)$. Let SPARSE denote the class of sparse sets. Then we have the following theorem.

Theorem 3.6 *All sparse sets in NP are PP-low. Namely, PP(SPARSE ∩ NP) = PP.*

The proof is quite different from the above one but is based on a slightly complecated simulation technique. It will appears in [6]. Let BPP denote the class of sets accepted by polynomial time-bounded probabilistic Turing machines with two-sided bounded error probability.

Theorem 3.7 *BPP is PP-low. Namely, PP(BPP) = PP.*

The proof of this also appears in [6].

4 Concluding remarks.

In this paper, we considered PP-computations relativized with oracle sets from some interesting classes and we showed that all informations from the oracle sets can be decided within PP itself. Our proof techniques are based on some simulations of the relativized machines by unrelativized machines. Hence we can observe that all equality mentioned in the previous section can be relativized with all oracle sets. That is, for all oracle sets A, PP(FewP(A)) = PP(SPARSE ∩ NP(A)) = PP(BPP(A)) = PP(A). From this observation, we observe that more complex classes are PP-low. For example, for any PP-low class C, FewP(C), SPARSE∩NP(C) and BPP(C) are PP-low. However, most important question whether NP is PP-low or not has remained open. If one could settle this question by using a relativizable technique, then he will observe that PH ⊆ PP, a glorious inclusion of PH in PP. Conversely, if one could show a good evidence such that PP(NP) is strictly harder than PP, then he must know that UP and FewP differ from NP. Then it is an interesting question whether there exists an oracle set separating PP(NP) from PP. Finally, we note that a lot of works closely related to this paper were recently done in [2,9,10] and that all results in this paper will appears in [6].

Acknowledgement The author would like to thank Osamu Watanabe for his many discussions on this work.

References

[1] E. Allender. The complexity of sprse sets in P. In *the 1st SICT conference, LNCS 223*, pages 1–11, 1986.

[2] R. Beigel, L. A. Hemachandra, and G. Wechsung. On the power of probabilistic polynomial-time: $P^{NP[log]} \subseteq PP$. 4th SICT conference, to appear, 1989.

[3] J. Gill. Computational complexity of probanilistic Turing machines. *SIAM J. Computing*, 6:675–695, 1977.

[4] L. A. Hemachandra. On ranking. In *the 2nd SICT conference*, pages 103–117, 1987.

[5] K. Ko and U. Shöning. On circuit-size complexity and the low hierarchy in NP. *SIAM J. Computing*, 14:41–51, 1985.

[6] J. Köbler, U. Shöning, S. Toda, and J. Toran. Turing machines with few accepting computations. 4th SICT conference, to appear, 1989.

[7] J. L. Balcázar, R. V. Book, and U. Shöning. Sparse sets, lowness, and highness. *SIAM J. Computing*, 15:739–747, 1986.

[8] U. Shöning. A low and a high hierarchy within NP. *J. Comput. Sys. Sci.*, 27:14–28, 1983.

[9] S. Toda. PP is $\leq_T^P$–hard for the polynomial-time hierarchy. *Submitted for pubilication*, 1989.

[10] S. Toda and O. Watanabe. On the power of #P functions. manuscript, 1989.

[11] J. Toran. An oracle characterization of the counting hierarchy. In *the 3rd SICT conference*, pages 213–223, 1988.

[12] K. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

[13] O. Watanabe. On hardness of one–way functions. *Imfor. Process. Let.*, 27:151–157, 1988.