Electronic Colloquium on Computational Complexity, Report No. 18 (2018)

Polynomial-Time Random Oracles and Separating Complexity Classes

John M. Hitchcock
Department of Computer Science
University of Wyoming
jhitchco@cs.uwyo.edu

Adewale Sekoni
Department of Computer Science
University of Wyoming
asekoni@uwyo.edu

Hadi Shafei
Department of Mathematics and Computer Science
Northern Michigan University
hshafei@nmu.edu

Abstract

Bennett and Gill (1981) showed that $\mathrm{P}^A \neq \mathrm{NP}^A \neq \mathrm{coNP}^A$ for a random oracle A, with probability 1. We investigate whether this result extends to individual polynomial-time random oracles. We consider two notions of random oracles: p-random oracles in the sense of martingales and resource-bounded measure (Lutz, 1992; Ambos-Spies et al., 1997), and p-betting-game random oracles using the betting games generalization of resource-bounded measure (Buhrman et al., 2000). Every p-betting-game random oracle is also p-random; whether the two notions are equivalent is an open problem.

(1) We first show that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every oracle A that is p-betting-game random.

Ideally, we would extend (1) to p-random oracles. We show that answering this either way would imply an unrelativized complexity class separation:

(2) If $\mathrm{P}^A \neq \mathrm{NP}^A$ relative to every p-random oracle A, then $\mathrm{BPP} \neq \mathrm{EXP}$.

(3) If $\mathrm{P}^A = \mathrm{NP}^A$ relative to some p-random oracle A, then $\mathrm{P} \neq \mathrm{PSPACE}$.

Rossman, Servedio, and Tan (2015) showed that the polynomial-time hierarchy is infinite relative to a random oracle, solving a longstanding open problem. We consider whether we can extend (1) to show that PH^A is infinite relative to oracles A that are p-betting-game random. Showing that PH^A separates at even its first level would also imply an unrelativized complexity class separation:

(4) If $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for a p-betting-game measure 1 class of oracles A, then $\mathrm{NP} \neq \mathrm{EXP}$.

(5) If PH^A is infinite relative to every p-random oracle A, then $\mathrm{PH} \neq \mathrm{EXP}$.

1 Introduction

Bennett and Gill [3] initiated the study of random oracles in computational complexity, proving that $\mathrm{P}^A \neq \mathrm{NP}^A$ for a random oracle A, with probability 1. Subsequent work showed that this holds for *individual* random oracles. Book, Lutz, and Wagner [4] showed that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every oracle

ISSN 1433-8092

A that is algorithmically random in the sense of Martin-Löf [14]. Lutz and Schmidt [13] improved this further to show $\mathrm{P}^A \neq \mathrm{NP}^A$ for every oracle A that is pspace-random [11].

We investigate whether this extends to individual polynomial-time random oracles [11, 2]. To show that $\mathrm{P}^A \neq \mathrm{NP}^A$ for p-random oracles A, we need to show that if $\mathrm{P}^A = \mathrm{NP}^A$, then there is a polynomial-time martingale that succeeds on A. This means that if A makes $\mathrm{P}^A = \mathrm{NP}^A$, then A is somehow predictable or simple.

Allender and Strauss [1] proved that $\{A \mid \mathrm{P}^A \neq \mathrm{BPP}^A\}$ has p-measure 0, which implies that $\mathrm{P}^A = \mathrm{BPP}^A$ for every p-random oracle A. This strengthens another result of Bennett and Gill [3] that $\mathrm{P}^A = \mathrm{BPP}^A$ holds for a random oracle A, with probability 1. Allender and Strauss's proof relies on derandomization [17] and is a different approach than Bennett and Gill. For P vs NP oracles, the best known is the pspace-randomness result of Lutz and Schmidt [13]. In related work, Kautz and Miltersen [10] showed that if A is an algorithmically random oracle, then NP^A does not have p-measure 0. Because the class $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has Hausdorff dimension 1 [8], there is a fundamental limit to how strongly a martingale can succeed on the class.

Each oracle A is associated with a *test language* L_A. This language is tally and $0^n \in L_A$ if and only if in the 2^n tribes of n strings following 0^n, there is at least one tribe contained in A. (See Section 3 for a precise definition of L_A. Bennett and Gill used a slightly different, but equivalent formulation of the test language.) It is clear that $L_A \in \mathrm{NP}^A$. From [3], we know that $\{A \mid L_A \in \mathrm{P}^A\}$ has Lebesgue measure 0. Since $\mathrm{P}^A = \mathrm{NP}^A$ implies $L_A \in \mathrm{P}^A$, it follows that $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has measure 0. We would like to show $\{A \mid L_A \in \mathrm{P}^A\}$ has p-measure 0.

Intuitively, if $L_A \in \mathrm{P}^A$, we would like to predict membership of strings in A. This would be relatively simple if the P^A algorithm asked only nonadaptive queries. However, since the queries may be adaptive, there are potentially exponentially many queries – too many to be considered by a polynomial-time martingale.

The difficulty is martingales are forced to bet on strings in lexicographic order. Buhrman et al. [5] introduced an extension of resource-bounded measure using *betting games*. Betting games are similar to martingales but they may adaptively choose the order in which they bet on strings. Whether betting games are equivalent to martingales is an open question [5]. The adaptiveness in betting games allows us to simulate P^A algorithms. We show in Section 3 that there is a p-betting game succeeding on $\{A \mid L_A \in \mathrm{P}^A\}$. Therefore $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-betting-game random oracle A.

In Section 4, we consider whether there are limitations to extending the betting games result. We show that determining whether or not $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has polynomial-time measure 0 (with respect to martingales) would imply a separation of complexity classes:

- If $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has p-measure 0, then BPP $\neq$ EXP.
- If $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ does not have p-measure 0, then P $\neq$ PSPACE.

This shows that determining the p-measure of $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$, or resolving whether $\mathrm{P}^A \neq \mathrm{NP}^A$ for all p-random A, is likely beyond current techniques.

Bennett and Gill [3] also showed that $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for a random oracle A, with probability 1. Rossman, Servedio, and Tan [18] answered a longtime open question [7] by extending Bennett and Gill's result to separate every level of the polynomial-time hierarchy. They proved an average case depth hierarchy theorem for Boolean circuits which implies that the polynomial-time hierarchy is infinite relative to a random oracle. Can we show that PH is infinite relative to polynomial-time random oracles as well? We show that extending our main result to separate PH^A at even the first level would separate NP from EXP:

- If $\{A \mid \mathrm{NP}^A = \mathrm{coNP}^A\}$ has p-betting-game measure 0, then $\mathrm{NP} \neq \mathrm{EXP}$.
- If PH^A is infinite relative to every p-random oracle A, then $\mathrm{PH} \neq \mathrm{EXP}$.

2 Preliminaries

We use standard notation. The binary alphabet is $\Sigma = \{0, 1\}$, the set of all binary strings is Σ^*, the set of all binary strings of length n is Σ^n, and the set of all infinite binary sequences is Σ^∞. The empty string is denoted by λ. We use the standard enumeration of strings, $s_0 = \lambda, s_1 = 0, s_2 = 1, s_3 = 00, s_4 = 01, \ldots$, and the standard lexicographic ordering of strings corresponds to this enumeration. The characteristic sequence of a language A is the sequence $\chi_A \in \Sigma^\infty$, where $\chi_A[n] = 1 \iff s_n \in A$. We refer to $\chi_A[s_n] = \chi_A[n]$ as the characteristic bit of s_n in A. A language A can alternatively be seen as a subset of Σ^*, or as an element of Σ^∞ via identification with its characteristic sequence χ_A. Given strings x, y we denote by $[x, y]$ the set of all strings z such that $x \leq z \leq y$. For any string s_n and number k, $s_n + k$ is the string s_{n+k}; e.g. $\lambda + 4 = 01$. Similarly we denote by $A[x, y]$ the substring of the characteristic sequence χ_A that corresponds to the characteristic bits of the strings in $[x, y]$.

2.1 Martingales and Betting Games

We now give a brief overview of martingales and betting games, and how they are applied in computational complexity to define resource-bounded measures and randomness notions. For further details, we refer to [11, 12, 2, 5, 6].

Betting games, which are also called nonmonotonic martingales, originated in the field of algorithmic information theory. In that setting they yield the notion of Kolmogorov-Loveland randomness (generalizing Kolmogorov-Loveland stochasticity) [16, 15]. The concept was introduced to computational complexity by Buhrman et al. [5]. First, we recall the definition of a martingale:

Definition. A *martingale* is a function $d : \Sigma^* \to [0, \infty)$ such that for all $w \in \Sigma^*$, we have the following averaging condition:

$$d(w) = \frac{d(w0) + d(w1)}{2}.$$

Intuitively, a martingale is betting in order on the characteristic sequence of an unknown language. The martingale starts with finite initial capital $d(\lambda)$. The quantity $d(w)$ represents the current capital the martingale has after betting on the first $|w|$ bits of a sequence that begins with w. The quantities $\pi(w, 0) = d(w0)/2d(w)$ and $\pi(w, 1) = d(w1)/2d(w)$ represent the fraction of its current capital that the martingale is wagering on 0 and 1, respectively, being the next bit of the sequence. This next bit is revealed and the martingale has $d(w0) = 2\pi(w, 0)d(w)$ in the case of a 0 and $d(w1) = 2\pi(w, 1)d(w)$ in the case of a 1.

Betting games are a generalization of martingales and have the additional capability of selecting which position in a sequence, or equivalently, which string in a language, to bet upon next. A betting game is permitted to select strings in a nonmonotone order, that is, it may bet on longer strings, then shorter strings, then longer strings again (with the important restriction that it may not bet on the same string twice). Like martingales, betting games must also satisfy the averaging law, i.e. the average of the betting game's capital after betting on a string s when s belongs and when s doesn't belong to the language is the same as its capital before betting on s. We use the following definition of a betting game from [5].

Definition. A betting game G is an oracle Turing machine that maintains a "capital tape" and a "bet tape," in addition to its standard query tape and worktapes. The game works in rounds $i = 1, 2, 3, \ldots$ as follows. At the beginning of each round i, the capital tape holds a nonnegative rational number C_{i-1}. The initial capital C_0 is some positive rational number. G computes a query string x_i to bet on, a bet amount $B_i, 0 \leq B_i \leq C_{i-1}$, and a bet sign $b_i \in \{-1, +1\}$. The computation is legal so long as x_i does not belong to the set $\{x_1, \cdots, x_{i-1}\}$ of strings queried in earlier rounds. G ends round i by entering a special query state. For a given oracle language A, if $x_i \in A$ and $b_i = +1$, or if $x_i \notin A$ and $b_i = -1$, then the new capital is given by $C_i := C_{i-1} + B_i$, else by $C_i := C_{i-1} - B_i$. We charge M for the time required to write the numerator and denominator of the new capital C_i down. The query and bet tapes are blanked, and G proceeds to round $i + 1$.

It is easy to see from the above definition that b_i and B_i can easily be computed from the current capital $C_i := C_{i-1} + b_i B_i$ of the betting game. Therefore, we can equivalently define a betting game by describing the computation of the current capital C_i without explicitly specifying the computation of b_i and B_i. We do this because it is clearer and more intuitive to describe the computation of the current capital of the betting game presented in the next section.

Definition. If a betting game G earns unbounded capital on a language A (in the sense that for every constant c there is a point at which the capital exceeds c when betting on A), we say that G *succeeds on A*. The *success set* of a betting game G, denoted $S^\infty[G]$, is the set of all languages on which G succeeds. A betting game G *succeeds on* a class X of languages if $X \subseteq S^\infty[G]$.

By adding a resource bound Δ on the computation of a betting game or martingale, we get notions of resource-bounded measure on Σ^∞. For this paper the resource bounds we use are $\mathrm{p} = \mathrm{DTIMEF}(n^{O(1)})$, $\mathrm{p}_2 = \mathrm{DTIMEF}(2^{(\lg n)^{O(1)}})$, and $\mathrm{pspace} = \mathrm{DSPACEF}(n^{O(1)})$. We say a class $X \subseteq \Sigma^\infty$ has Δ-betting-game measure 0, if there is a Δ-computable betting game that succeeds on every language in it. It has Δ-measure 0 if the betting game is also a martingale [11]. A class X has Δ-betting-game measure 1 if X^c has Δ-betting-game measure 0. Similarly, X has Δ-measure 1 if X^c has Δ-measure 0. A language A is Δ-betting-game random if there is no Δ-computable betting game that succeeds on A. Similarly, A is Δ-random if there is no Δ-computable martingale that succeeds on A.

The ability of the betting game to examine a sequence nonmonotonically makes determining its running time complicated, since each language can induce a unique computation of the betting game. In other words, the betting game may choose to examine strings in different orders depending upon the language it is wagering against. Buhrman et al. looked at a betting game as an infinite process on a language, rather than a finite process on a string. They used the following definition:

Definition. A betting game G runs in time $t(2^n)$ if for all languages A, every query of length n made by G occurs in the first $t(2^n)$ steps of the computation.

Specifically, once a $t(2^n)$-time-bounded betting game uses $t(2^n)$ computational steps, it cannot go back and select any string of length n. Most importantly, no polynomial-time betting game can succeed on the class $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$.

2.2 Martingales and Betting Games: Intuitive view

Intuitively, a betting game can be viewed as the strategy of a gambler who bets on infinite sequence of strings. The gambler starts with initial capital C, then begins to query strings to bet on. The gambler's goal is to grow the capital C without bound. The same view holds for martingales with the restriction that the gambler must bet on the strings in the standard ordering.

3 Betting Game Random Oracles

In this section we show that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-betting-game random oracle.

Theorem 3.1. *The class $\{A \mid \mathrm{P}^A \neq \mathrm{NP}^A\}$ has* p*-betting-game measure 1. In particular, $\mathrm{P}^A \neq \mathrm{NP}^A$ for every* p*-betting-game random oracle A.*

Proof. Given a language A we define the test language

$$L_A = \{0^n \mid \mathrm{Tribes}_{2^n,n}(A[0^n + 1, 0^n + n2^n]) = 1\},$$

where $\mathrm{Tribes}_{2^n,n} : \{0,1\}^{n2^n} \longrightarrow \{0,1\}$ is defined as follows. Given $w \in \{0,1\}^{n2^n}$, first we view w as concatenation of 2^n length n strings $w_1, w_2, \cdots, w_{2^n}$; i.e. $w = w_1 w_2 \cdots w_{2^n}$. $\mathrm{Tribes}_{2^n,n}(w)$ is 1 if and only if $w_i = 1^n$ for some i. Secondly, we view w as the substring $A[0^n + 1, 0^n + n2^n]$ of the characteristic sequence of some language A. With both views in mind, we define a tribe to be the set of strings whose characteristic bits are encoded by some w_i. For example, given any $i \in [1, 2^n]$, the set of strings $[0^n + (i-1)n + 1, 0^n + in]$ is a tribe because its characteristic bits are encoded by w_i. Since the n strings in any tribe have length $O(n)$, an NP oracle machine can easily verify the membership of any 0^n, therefore $L_A \in \mathrm{NP}^A$. Now we define a betting game G that succeeds on the set $X = \{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$, thereby proving the theorem. Our betting game G is going to simulate oracle Turing machines on some strings in the set $\{0^n \mid n \in \mathbb{N}\}$. Let $M_1, M_2, \cdots$ be an enumeration of all oracle TMs, where M_i runs in time at most $n^{\lg i} + i$ on inputs of length n. The initial capital of G is 2 and we view it as composed of infinite "shares" $a_i = b_i = 2^{-i}, i \in \mathbb{N}$ that are used by G to bet on some of the strings it queries.

Before we go into the details of the implementation of G, we give a high level view. The strategy of G to succeed on X is quite simple. For any language A, the cardinality of $\{0^n \mid 0^n \notin L_A\}$ is either finite or infinite. When it is finite, after seeing a finite number of strings all following strings will belong to L_A. G uses "shares" a_i reserved at its initialization to bet in this situation. On the other hand when it is infinite and $A \in X$ we can find an oracle TM M_i that decides L_A. Most importantly this TM rejects its input infinitely often and it is only in this situation that we bet with the b_i "shares". Details follow.

First we specify the order in which G queries strings followed by which strings it bets on. G operates sequentially in stages $1, 2, \cdots$. In stage j, G queries 0^{n_j}, where n_j is the smallest integer such that all the strings queried in stage $j-1$ have length less than n_j. G then runs the oracle TM M_{i+1} on 0^{n_j}, where i is the number of TMs simulated in the previous stages whose output was inconsistent with L_A in one of the previous stages. During the simulation of M_{i+1}, G answers any queries made by the TM either by looking up the string from its history, or if the string isn't in its history, then G queries it. After the simulation, G queries in the standard lexicographic order all the strings in the 2^{n_j} tribes that follow 0^{n_j} that haven't already been queried. Finally, to complete stage j, G queries all the remaining strings of length at most the length of the longest string queried by G so far.

Now we specify which strings G bets on and how it bets with the a_i's and b_i's. In stage j, let i and n_j be such that M_i is the Turing machine simulated in this stage and 0^{n_j} is the input it will be simulated on. The only strings G bets on will be the $n_j 2^{n_j}$ strings following 0^{n_j}; i.e. the tribes. We use a_l and b_i, two of the infinite "shares" of our initial capital reserved by G for betting, where l is the smallest positive integer such that $a_l \neq 0$. As will be shown later we do this because G loses all of a_l whenever $0^n \notin L_A$. The "shares" a_l and b_i are dynamic and may have their values updated as we bet with them. Therefore, the current capital of G after each bet is $\sum_{i=1}^{\infty}(a_i + b_i)$. Though we describe separately how G bets with a_l and b_i, we may bet with both simultaneously. We bet with some a_l for every stage, but with the b_i's we bet only when the output of the simulated TM is 0. Therefore every time we bet with b_i we also simultaneously bet with a_l. First let us see how G bets in stage j using the a_l and then with b_i.

Betting with a_l: Our choice of l ensures that $a_l \neq 0$. In fact, a_l will either increase, or reduce to 0 after betting. If we lose a_l in the current stage, then we use $a_{l+1} = 2^{-(l+1)}$ to bet in the next stage. G uses a_l to bet that at least one of the 2^{n_j} tribes that follow 0^{n_j} is completely contained in A; i.e. $0^{n_j} \in L_A$. Call this event $\mathcal{B}_{n_j}$. It is easy to see that for sufficiently large n_j, when strings are included independently in A with probability 1/2, the probability of event $\mathcal{B}_{n_j}$ is

$$\mathrm{P}_r(\mathcal{B}_{n_j}) = 1 - (1 - 2^{-n_j})^{2^{n_j}} \approx 1 - 1/e.$$

G bets in such a way that whenever the sequence of strings seen satisfies the event $\mathcal{B}_{n_j}$, a_l increases by a factor of approximately $1/(1 - 1/e)$. If the sequence of strings does not satisfy event $\mathcal{B}_{n_j}$ then G loses all of a_l and will bet with a_{l+1} in the next stage.

We now elaborate on how a_l increases by a factor of approximately $1/(1-1/e)$ when event $\mathcal{B}_{n_j}$ occurs. Let $\omega \in \{0, 1, \star\}^{n_j 2^{n_j}}$ represent the current status of strings in $[0^{n_j} + 1, 0^{n_j} + n_j 2^{n_j}]$, $\omega[i]$ indicates the status of string $0^{n_j} + i$, $\star$ indicates the string has not been queried by G yet, and bits 0 and 1 have their usual meaning. Define

$$G_{a_l}(\omega) = \frac{a_l}{\Pr(\mathcal{B}_{n_j})} \Pr(\mathcal{B}_{n_j} | \omega),$$

where $\Pr(\mathcal{B}_{n_j})$ is the probability a random language satisfies event $\mathcal{B}_{n_j}$, and $\Pr(\mathcal{B}_{n_j}|\omega)$ is the conditional probability of the event $\mathcal{B}_{n_j}$ given the current status of the strings as encoded by ω, i.e. given the strings in $[0^{n_j} + 1, 0^{n_j} + n_j 2^{n_j}]$ whose membership in A has already been revealed, what is the probability that randomly assigning membership to other strings causes event $\mathcal{B}_{n_j}$ to occur. This probability is rational and easy to compute in $O(2^{2n})$ time by examining the status of the strings in each of the 2^n tribes in $[0^n + 1, 0^n + n2^n]$. G_{a_l} is essentially a martingale. Whenever the membership of any string in $[0^{n_j} + 1, 0^{n_j} + n_j 2^{n_j}]$ is revealed, a_l is then updated to $G_{a_l}(\omega)$. Given $\omega \in \{0, 1, \star\}^{n_j 2^{n_j}}$ and $b \in \{0, 1, \star\}$, let $\omega^{i \to b}$ denote ω with its i^{th} symbol set to b. It is easy to see that

$$G_{a_l}(\omega^{i \to \star}) = \frac{G_{a_l}(\omega^{i \to 0}) + G_{a_l}(\omega^{i \to 1})}{2}.$$

For all sufficiently large n_j,

$$G_{a_l}(\omega) = \frac{a_l}{\mathrm{P}_r(B_{n_j})} \approx a_l/(1 - 1/e)$$

for any string $\omega \in \{0,1\}^{n_j 2^{n_j}}$ that satisfies event $\mathcal{B}_{n_j}$ and 0 for those that do not satisfy $\mathcal{B}_{n_j}$. It is important to note that G can always bet with a_l no matter the order in which it requests the strings in $[0^{n_j}+1, 0^{n_j}+n_j 2^{n_j}]$ that it bets on. But as will be shown next the ordering of these strings is important when betting with b_i.

Betting with b_i: Finally, we specify how G bets with "share" b_i which is reserved for betting with M_i. G only bets with b_i when the simulation of M_i on 0^{n_j} returns 0. In this situation G bets that at least $2^{n_j} - (n_j^{\lg i} + i)$ tribes of the 2^{n_j} tribes that follow 0^{n_j} are not contained in A. For simplicity, G does not bet on the tribes that M_i queried. We denote by $\mathcal{C}_{n_j}$ the event that all the tribes not queried by M_i are not contained in A. Event $\mathcal{C}_{n_j}$ occurs with probability at most $(1-2^{-n_j})^{2^{n_j}-(n_j^{\lg i}+i)} \approx 1/e$, and is almost the complement of $\mathcal{B}_{n_j}$. In this case G bets similarly to how it bets with a_l and increases b_i by a factor of $1/\mathrm{P}_r(\mathcal{C}_{n_j}) \approx e$ whenever the sequence of strings that follow 0^{n_j} satisfies $\mathcal{C}_{n_j}$. If the sequence does not satisfy $\mathcal{C}_{n_j}$ then G loses all of b_i.

We now argue that G succeeds on X. Suppose $A \in X$ and $S \subseteq 0^*$ is the set of input strings G simulates on some TMs in stages $1, 2, \ldots$. Then there are two possibilities:

1. Finitely many strings in S do not belong to L_A,

2. Infinitely many strings in S do not belong to L_A.

Denote by s_k the k^{th} string in S. In the first case, there must be a k such that for every stage $j \geq k$, $s_j \in L_A$. Once we reach stage k, G uses a "share" of its capital $a_i \neq 0$ to bet on s_j belonging to L_A for all $j \geq k$. Therefore G will increase a_i by a factor of approximately $1/(1-1/e)$ for all but finitely many stages $j \geq k$. Therefore the capital of G will grow without bound in this case.

In the second case, we must reach some stage k at which we use the correct oracle TM M_i that decides L_A on inputs in S. From this stage onward G will never change the TM it simulates on the strings in S we have not seen yet. In this case we are guaranteed this simulation will output 0 infinitely often. It follows by the correctness of M_i and the definition of G that whenever the output of M_i is 0 the "share" of the capital b_i reserved for betting on M_i will be increased by a factor of approximately e. Since this condition is met infinitely often, it follows that the capital of G increases without bound in this case also.

Finally we show that G can be implemented as a $O(2^{2n})$-betting game; i.e. after $O(2^{2n})$ time, G will have queried all strings of length n. First, we bound the runtime of each round of the betting game; i.e. the time required to bet on a string. This should not be confused with the stages of G which include several rounds of querying. In each round, we have to compute $\sum_{i=1}^{\infty}(a_i + b_i)$ the current capital of G. This sum can easily be computed in $O(2^n)$ time. This is because for each round we change at most two "shares" a_l and b_i to some rational numbers that can be computed in $O(2^n)$ time. The remaining a and b "shares" with indices less than i and l respectively have values 0 and those with indices grater than i and l respectively retain their initial values. We may also simulate a TM in each round. Since each simulated TM M_i has $i \leq n$ it takes $O(n^{\lg n})$ time for the simulation of M_i on 0^n. Therefore, each round is completed in $O(2^n)$ time. After the simulation G requests all the remaining strings in $[0^n+1, 0^n+n2^n]$ that were not queried during the simulation. Therefore, it takes $O(2^{2n})$ time for G to have requested all strings of length n. $\square$

4 Limitations

In this section we examine the possibility of extending Theorem 3.1. We show that it cannot be improved to p-random oracles or improved to separate the polynomial-time hierarchy without separating BPP or NP from EXP, respectively. On the other hand, showing that Theorem 3.1 cannot be improved to p-random oracles would separate PSPACE from P.

4.1 Does $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-random oracle A?

We showed in Theorem 3.1 that $\mathrm{P}^A \neq \mathrm{NP}^A$ for a p-betting-game random oracle. It is unknown whether p-betting games and p-martingales are equivalent. If they are, then BPP $\neq$ EXP [5]. This is based on the following theorem and the result that $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete languages for EXP have p-betting-game measure 0 [5].

Theorem 4.1 (Buhrman et al. [5])**.** *If the class of* $\leq_{\mathrm{T}}^{\mathrm{P}}$*-complete languages for* EXP *has* p_2*-measure zero then* BPP $\neq$ EXP.

We show improving Theorem 3.1 to p-random oracles would also imply BPP $\neq$ EXP. First, we prove the following for p_2-measure.

Theorem 4.2. *If* $\{A \mid \mathrm{P}^A \neq \mathrm{NP}^A\}$ *has* p_2*-measure 1, then* BPP $\neq$ EXP.

Proof. If L is any $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete language for EXP, then

$$\mathrm{NP}^L \subseteq \mathrm{EXP} \subseteq \mathrm{P}^L \subseteq \mathrm{NP}^L.$$

Therefore the class of $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete languages for EXP is a subset of $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$. If $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has p_2-measure 0 then so does the class of $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete languages of EXP. Theorem 4.1 implies that BPP $\neq$ EXP. □

We have the following for p-random oracles by the universality of p_2-measure for p-measure [11].

Corollary 4.3. *If* $\mathrm{P}^A \neq \mathrm{NP}^A$ *for every* p*-random oracle* A*, then* BPP $\neq$ EXP.

Proof. The hypothesis implies that every A with $\mathrm{P}^A = \mathrm{NP}^A$ is not p-random, i.e. there is a p-martingale that succeeds on A. Let d' be a p_2-martingale d' that is universal for all p-martingales [11]: $S^\infty[d] \subseteq S^\infty[d']$ for every p-martingale d. Then d' succeeds on $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$. □

4.2 Is it possible that $\mathrm{P}^A = \mathrm{NP}^A$ for some p-random oracle A?

Given Theorem 4.2, we consider the possibility of whether $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ does not have p-measure 0. Because Lutz and Schmidt [13] showed that this class has pspace-measure 0, it turns out that if it does not have p-measure 0, then we have a separation of PSPACE from P.

Theorem 4.4 (Lutz and Schmidt [13])**.** *The class* $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ *has* pspace*-measure 0.*

We note that because every p-betting game may be simulated by a pspace-martingale [5], Theorem 4.4 follows as a corollary to Theorem 3.1.

Lemma 4.5. *If* P = PSPACE*, then for every* pspace*-martingale* d*, there is a* p*-martingale* d' *with* $S^\infty[d] \subseteq S^\infty[d']$*.*

Proof. Let $d : \{0,1\}^* \longrightarrow [0,\infty)$ be a pspace-martingale. Without loss of generality also assume that d is exactly computable [9] and its output is in $\{0,1\}^{\leq p(n)}$, for some polynomial p. Consider the language $L_d = \{\langle w, i, b\rangle \mid \text{the } i\text{th bit of } d(w) \text{ is } b\}$. Clearly $L_d \in \mathrm{PSPACE}$ and hence also in P by our hypothesis. We can therefore compute $d(w)$ is polynomial time using L_d. □

Theorem 4.6. *If* $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ *does not have* p*-measure 0, then* $\mathrm{P} \neq \mathrm{PSPACE}$*.*

Proof. Assume $\mathrm{P} = \mathrm{PSPACE}$. Theorem 4.4 and Lemma 4.5 imply that $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has p-measure 0. □

Corollary 4.7. *If there is a* p*-random oracle* A *such that* $\mathrm{P}^A = \mathrm{NP}^A$*, then* $\mathrm{P} \neq \mathrm{PSPACE}$*.*

4.3 Is PH infinite relative to p-betting-game random oracles?

Bennett and Gill [3] showed that $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for a random oracle A, with probability 1. Thus PH^A does not collapse to its first level. Rossman, Servedio, and Tan [18] showed that PH^A is infinite relative to a random oracle, with probability 1.

Can we improve Theorem 3.1 to show that PH^A does not collapse for a p-betting-game random oracle? This also has complexity class separation consequences:

Theorem 4.8. *For* $k > 0$*, let* $X_k = \{A \mid \sum_k^{\mathrm{P},A} = \prod_k^{\mathrm{P},A}\}$*. If* X_k *has* p_2*-betting-game measure zero, then* $\sum_k^{\mathrm{P}} \neq \mathrm{EXP}$*.*

Proof. We prove the contrapositive. Suppose $\sum_k^{\mathrm{P}} = \mathrm{EXP}$, then $\prod_k^{\mathrm{P}} = \mathrm{EXP}$. Given $A \in \mathrm{EXP}$, then the following containments hold:

$$\Sigma_k^{\mathrm{P}} \subseteq \Sigma_k^{\mathrm{P},A} \subseteq \mathrm{EXP} = \Pi_k^{\mathrm{P}} \subseteq \Pi_k^{\mathrm{P},A} \subseteq \mathrm{EXP} = \Sigma_k^{\mathrm{P}}.$$

turn implies that $\mathrm{EXP} \subseteq X_k$. Since EXP does not have p_2-betting-game measure zero [5] then neither does X_k. Hence, the Theorem follows. □

In particular, we have the following for the first level of PH:

Corollary 4.9. *If* $\{A \mid \mathrm{NP}^A \neq \mathrm{coNP}^A\}$ *has p-betting-game measure 1, then* $\mathrm{NP} \neq \mathrm{EXP}$*.*

Because it is open whether betting games have a union lemma [5], it is not clear whether Corollary 4.9 may be extended to show that if $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for every p-betting-game random oracle A, then $\mathrm{NP} \neq \mathrm{EXP}$. This extension would hold if there is a p_2-betting game that is universal for all p-betting games. However, we do have the following for p-random oracles.

Corollary 4.10. *If* $\mathrm{NP}^A \neq \mathrm{coNP}^A$ *for every* p*-random oracle* A*, then* $\mathrm{NP} \neq \mathrm{EXP}$*.*

Corollary 4.11. *If* PH^A *is infinite for every* p*-random oracle* A*, then* $\mathrm{PH} \neq \mathrm{EXP}$*.*

5 Conclusion

We have shown that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-betting-game random oracle A (Theorem 3.1). Establishing whether this also holds for p-random oracles would imply either $\mathrm{BPP} \neq \mathrm{EXP}$ (Corollary 4.3) or $\mathrm{P} \neq \mathrm{PSPACE}$ (Corollary 4.7). These results, together with Theorems 4.4 and 4.8, motivate investigating the status of PH relative to pspace-random oracles. In particular:

1. Does $\{A \mid \mathrm{NP}^A = \mathrm{coNP}^A\}$ have pspace-measure 0?

2. More generally, does $\{A \mid \mathrm{PH}^A \text{ collapses}\}$ have pspace-measure 0?

References

[1] E. Allender and M. Strauss. Measure on small complexity classes with applications for BPP. In *Proceedings of the 35th Symposium on Foundations of Computer Science*, pages 807–818. IEEE Computer Society, 1994.

[2] K. Ambos-Spies and E. Mayordomo. Resource-bounded measure and randomness. In A. Sorbi, editor, *Complexity, Logic and Recursion Theory*, Lecture Notes in Pure and Applied Mathematics, pages 1–47. Marcel Dekker, New York, N.Y., 1997.

[3] C. H. Bennett and J. Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1. *SIAM Journal on Computing*, 10:96–113, 1981.

[4] R. V. Book, J. H. Lutz, and K. W. Wagner. An observation on probability versus randomness with applications to complexity classes. *Mathematical Systems Theory*, 27:201–209, 1994.

[5] H. Buhrman, D. van Melkebeek, K. W. Regan, D. Sivakumar, and M. Strauss. A generalization of resource-bounded measure, with application to the BPP vs. EXP problem. *SIAM Journal on Computing*, 30(2):576–601, 2001.

[6] R. C. Harkins and J. M. Hitchcock. Exact learning algorithms, betting games, and circuit lower bounds. *ACM Transactions on Computation Theory*, 5(4):article 18, 2013.

[7] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[8] J. M. Hitchcock. Hausdorff dimension and oracle constructions. *Theoretical Computer Science*, 355(3):382–388, 2006.

[9] D. W. Juedes and J. H. Lutz. Weak completeness in E and E_2. *Theoretical Computer Science*, 143(1):149–158, 1995.

[10] S. M. Kautz and P. B. Miltersen. Relative to a random oracle, NP is not small. *Journal of Computer and System Sciences*, 53(2):235–250, 1996.

[11] J. H. Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and System Sciences*, 44(2):220–258, 1992.

[12] J. H. Lutz. The quantitative structure of exponential time. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 225–254. Springer-Verlag, 1997.

[13] J. H. Lutz and W. J. Schmidt. Circuit size relative to pseudorandom oracles. *Theoretical Computer Science*, 107(1):95–120, March 1993.

[14] P. Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[15] W. Merkle, J. S. Miller, A. Nies, J. Reimann, and F. Stephan. Kolmogorov-Loveland randomness and stochasticity. *Annals of Pure and Applied Logic*, 138(1–3):183–210, 2006.

[16] A. A. Muchnik, A. L. Semenov, and V. A. Uspensky. Mathematical metaphysics of randomness. *Theoretical Computer Science*, 207(2):263 – 317, 1998.

[17] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[18] B. Rossman, R. A. Servedio, and L.-Y. Tan. An average-case depth hierarchy theorem for boolean circuits. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1030–1048. IEEE, 2015.