Electronic Colloquium on Computational Complexity, Report No. 196 (2010)

NE is not NP Turing Reducible to Nonexpoentially Dense NP Sets

Bin Fu*

Department of Computer Science
University of Texas–Pan American
Edinburgh, TX 78541, USA
Emails: binfu@cs.panam.edu

Abstract

A long standing open problem in the computational complexity theory is to separate NE from BPP, which is a subclass of $\mathrm{NP_T(NP)} \cap \mathrm{P/Poly}$. In this paper, we show that $\mathrm{NE} \not\subseteq \mathrm{NP_T(NP} \cap$ Nonexponentially-Dense-Class), where Nonexponentially-Dense-Class is the class of languages A without exponential density (for each constant $c > 0$, $|A^{\le n}| \le 2^{n^c}$ for infinitely many integers n). Our result implies $\mathrm{NE} \not\subseteq \mathrm{NP_T}(\mathrm{padding}(\mathrm{NP}, g(n)))$ for every time constructible super-polynomial function $g(n)$ such as $g(n) = n^{\lceil \log\lceil \log n\rceil\rceil}$, where $\mathrm{Padding}(\mathrm{NP}, g(n))$ is class of all languages $L_B = \{s10^{g(|s|)-|s|-1} : s \in B\}$ for $B \in \mathrm{NP}$. We also show $\mathrm{NE} \not\subseteq \mathrm{NP_T}(\mathrm{P}_{tt}(\mathrm{NP}) \cap \mathrm{TALLY})$.

1. Introduction

Separating the complexity classes has been one of the central problems in complexity theory. Separating NEXP from P/Poly is a long standing fundamental open problem in the computational complexity theory. We do not even know how to separate NEXP from BPP, which is a subclass of $\mathrm{NP_T(NP)} \cap \mathrm{P/Poly}$ proved by Adleman [1].

Whether sparse sets are hard for complexity classes plays an important role in the computational complexity theory (for examples, [3, 15, 17, 19]). It is well known that P/Poly is the same as the class of languages that are truth table reducible to tally sets ($\mathrm{P/Poly} = \mathrm{P}_{tt}(\mathrm{TALLY})$). The combination of bounded number of queries and density provides an approach to characterize the complexity of the nonuniform computation models. The partial progress for separating exponential time classes from nonuniform polynomial time classes are shown in [8, 11, 13, 16, 21]. Let Nonexponentially-Dense-Class be the class of languages A without exponential density (for each constant $c > 0$, $|A^{\le n}| \le 2^{n^c}$ for infinitely many integers n). Improving Hartmanis and Berman's separation $\mathrm{E} \not\subseteq \mathrm{P}_m$(Nonexponentially-Dense-Class) [3], Watanabe showed $\mathrm{E} \not\subseteq \mathrm{P}_{btt}$(Nonexponentially-Dense-Class). Watanabe's result was improved by two research groups independently with incomparable results that $\mathrm{E} \not\subseteq \mathrm{P}_{n^{1-\epsilon}-tt}$(Nonexponentially-Dense-Class) by Lutz and Mayordomo [16], and $\mathrm{EXP} \not\subseteq \mathrm{P}_{n^{1-\epsilon}-\mathrm{T}}$(Nonexponentially-Dense-Class) and $\mathrm{E} \not\subseteq \mathrm{P}_{n^{\frac{1}{2}-\epsilon}-\mathrm{T}}$(Nonexponentially-Dense-Class) by Fu [8]. Fu's results were improved to $\mathrm{E} \not\subseteq \mathrm{P}_{n^{1-\epsilon}-\mathrm{T}}$(Nonexponentially-Dense-Class) by Hitchcock [13]. A recent celebrated progress was made by Williams separating NEXP from ACC [22]. It is still an open problem to separate NEXP from $\mathrm{P}_{O(n)-tt}(\mathrm{TALLY})$.

The nondeterministic time hierarchy was separated in the early research of complexity theory by Cook [7], Serferas, Fischer, Meyer [20], and Zak [23]. A separation with immunity among nondeterministic computational complexity classes was derived by Allender, Beigel, Hertranpf and Homer [2]. The difference between NE and NP has not been fully solved. One of the most interesting problems between them is to separate NE from $\mathrm{P_T(NP)}$. Fu, Li and Zhong [10] showed $\mathrm{NE} \not\subseteq \mathrm{P}_{n^{o(1)}-\mathrm{T}}(\mathrm{NP})$. Their result was later improved by Mocas [18] to $\mathrm{NEXP} \not\subseteq \mathrm{P}_{n^c-\mathrm{T}}(\mathrm{NP})$ for any constant $c > 0$. Mocas's result is optimal with respect to relativizable proofs, as Buhrman and Torenvliet [5] showed an oracle relative to which $\mathrm{NEXP} = \mathrm{P_T(NP)}$. Buhrman,

*This research is supported in part by National Science Foundation Early Career Award 0845376.

ISSN 1433-8092

Fortnow and Santhanam [4] and Fu, Li and Zhang [9] showed NEXP $= \mathrm{P}_{n^c-\mathrm{T}}(\mathrm{NP})/n^c$ for every constant $c > 0$ (two papers appeared in two conferences with a similar time). Fu, Li and Zhang showed that NEXP is not reducible to tally sets by the polynomial time nondeterministic Turing reductions with the number of queries bounded by a sub-polynomial function $g(n)$ such as $g(n) = n^{\frac{1}{\log\log n}}$ (NE $\not\subseteq \mathrm{NP}_{g(n)-\mathrm{T}}(\mathrm{TALLY})$)[9].

In this paper, we show that NE $\not\subseteq \mathrm{NP}_{\mathrm{T}}(\mathrm{NP} \cap$ Nonexponentially-Dense-Class). Our result implies NE $\not\subseteq \mathrm{NP}_{\mathrm{T}}(\mathrm{padding}(\mathrm{NP}, g(n)))$ for every time constructible super-polynomial function $g(n)$ such as $g(n) = n^{\lceil \log \lceil \log n \rceil \rceil}$, where Padding(NP, $g(n)$) is the class of all languages $L_B = \{s10^{g(|s|)-|s|-1} : s \in B\}$ for $B \in$ NP. We also show NE $\not\subseteq \mathrm{NP}_{\mathrm{T}}(\mathrm{P}_{tt}(\mathrm{NP}) \cap \mathrm{TALLY})$.

This paper is organized as follows. Some notations are given in section 2. In section 3, we give a brief description of our method to prove the main result. In section 4, we separate NE from $\mathrm{NP}_{\mathrm{T}}(\mathrm{NP} \cap$ Nonexponentially-Dense-Class). In section 5, we show how to use the padding method to derive sub-exponential density problems in the class NP. In section 6, we separate NE from $\mathrm{NP}_{\mathrm{T}}(\mathrm{P}_{tt}(\mathrm{NP}) \cap \mathrm{TALLY})$. The conclusions are given in section 7.

2. Notations

Let $N = \{0, 1, 2, \cdots\}$ be the set of all natural numbers. Let $\Sigma = \{0, 1\}$ be the alphabet for all the languages in this paper. The length of a string s is denoted by $|s|$. Let A be a language. $A^{\leq n}$ is the subset of strings of length at most n in A. $A^{=n}$ is the subset of strings of length n in A. For a finite set X, let $|X|$ be the number of elements in X. For a Turing machine $M(.)$, let $L(M)$ be the language accepted by M. We use a pairing function $(.,.)$ with $|(x, y)| = O(|x| + |y|)$.

For a function $t(n) : N \to N$, let DTIME($t(n)$) be the class of languages accepted by deterministic Turing machines in $O(t(n))$ time, and NTIME($t(n)$) be the class of languages accepted by nondeterministic Turing machines in $O(t(n))$ time. Define the exponential time complexity classes: $\mathrm{E} = \cup_{c=1}^{\infty} \mathrm{DTIME}(2^{cn})$, $\mathrm{EXP} = \cup_{c=1}^{\infty} \mathrm{DTIME}(2^{n^c})$, $\mathrm{NE} = \cup_{c=1}^{\infty} \mathrm{NTIME}(2^{cn})$ and $\mathrm{NEXP} = \cup_{c=1}^{\infty} \mathrm{NTIME}(2^{n^c})$.

A language L is *sparse* if for some constant $c > 0$, $|L^{\leq n}| \leq n^c$ for all large n. Let SPARSE represent all sparse languages. Let TALLY be the class of languages with alphabet $\{1\}$.

Assume that $M(.)$ is an oracle Turing machine. A decision computation $M^A(x)$ returns either 0 or 1 when the input is x and oracle is A.

Let $\leq_r^{\mathrm{P}}$ be a type of polynomial time reductions, and S be a class of languages. $\mathrm{P}_r(S)$ is the class of languages A that are reducible to some languages to S via $\leq_r^P$ reductions. In particular, $\leq_m^{\mathrm{P}}$ is the polynomial time *many-one reduction*, and $\leq_{\mathrm{T}}^{\mathrm{P}}$ is the polynomial time *Turing reduction*.

For a class C of languages, we use $\mathrm{NP}_{\mathrm{T}}(C)$ to represent the class of languages that can be reducible to the languages in C via polynomial time nondeterministic Turing reductions.

For a nondecreasing function $d(n) : N \to N$, define Density($d(n)$) to be the class of languages A with $|A^{\leq n}| \leq d(n)$ for all sufficiently large n.

For a function $f(n) : N \to N$, it is *time constructible* if given n, $f(n)$ can be computed in $O(f(n))$ steps by a deterministic Turing machine.

A function $d(n) : N \to N$ is *nonexponential* if for every constant $c > 0$, $d(n) < 2^{n^c}$ for infinitely many integers n. Nonexponentially-Dense-Class is the class of languages A whose density function $d_A(n) = |A^{\leq n}|$ is nonexponential.

3. Overview of Our Method

We give a brief description about our method in this section. Our main theorem is proved by contradiction. Assume that NEXP $\subseteq \mathrm{NP}_{\mathrm{T}}(S)$, where S is a language in both NP and Nonexponentially-Dense-Class. Since S is not of exponential density, we can find a function nondecreasing unbounded function $e(1^n)$ that is computable in $2^{n^{O(1)}}$ time and satisfies $|S^{\leq n}| \leq 2^{n^{\frac{1}{e(1^n)^2}}}$ for infinitely many integers n. Let $h(n) = n^{e(1^n)}$. Thus, $h(n)$ is super-polynomial function.

Our main technical contribution is a counting method to be combined with the classical translational method in deriving the separation. Select an arbitrary language L_0 in DTIME($2^{h(n)}$). We define the language $L_1 = \{x10^{h(|x|)-|x|-1} : x \in L_0\}$. This converts L_0 into a language in NEXP. Using the assumption NEXP $\subseteq \mathrm{NP_T}(S)$, we have a polynomial time oracle Turing machine M_1 to accept L_1 with oracle S.

Define another language $L_2 = \{1^n 0m : m \le 2^n$ and there are at least m different strings $z_1, \cdots, z_m$ that are queried by M_1 with some input of length $h(n)\}$. We can also show that L_2 is also in NEXP. When S has a subexponential number of elements with length at most $h(n)^{O(1)}$, we show that the largest m with $1^n 0m \in L_2$ has $m < 2^n$.

In the next, we spend $2^{n^{O(1)}}$ time to find the largest m, which will be denoted by m_n. This can be easily done since L_2 is in $\mathrm{NP_T}(\mathrm{NP})$.

For m_n with $m_n < 2^n$, consider a nondeterministic computation that given an input (x, m_n) with $n = |x|$, it guesses all the strings $z_1, \cdots, z_{m_n}$, which are queried by M_1 by inputs of length $h(n)$, of S in a path. Thus, any query like $y \in S$? is identical to check if y is equal to one of elements in $z_1, \cdots, z_{m_n}$. This is an nondeterministic computation of exponential time. It can be converted into a problem in $\mathrm{NP_T}(\mathrm{NP})$. It can be simulated in a deterministic $2^{n^{O(1)}}$ time. Since there are infinitely many integers n with $|S^{\le n}| \le 2^{n^{\frac{1}{e(1^n)^2}}}$, we have infinitely many integers $n_1, n_2, \cdots$ to meet this case with $m_{n_i} < 2^{n_i}$. This brings a $2^{n^{O(1)}}$ time deterministic Turing machine M_* that $L_0^{=n_i} = L(M_*)^{=n_i}$ for some for infinitely many integers n_i. We can construct L_0 in $\mathrm{DTIME}(2^{h(n)})$ to make it impossible using the standard diagonal method. This brings a contradiction.

4. Main Separation Theorem

In this section, we present our main separation theorem. The theorem is achieved by the translational method, which is combined with a counting method to count the number of all possible strings queried by nondeterministic polynomial time oracle Turing machine.

Definition 1.

- Let M be an oracle nondeterministic Turing machine. Let $a_1 \cdots a_{i-1}$ be a $0,1$-sequence, and y be an input for M. Define $H(M(y), a_1 \cdots a_{i-1})$ to be the set of all strings z that are queried by $M(y)$ at the i-th time at some path assuming M receives answers $a_1, \cdots, a_{i-1}$ for its first $i-1$ queries from the oracle (the answer for each query is either '0' or '1' from the oracle).

- For a nondeterministic oracle Turing machine $M(.)$ and oracle A, and an integer k, define $Q(M, A, k)$ to be the set all strings z in A such that $z \in H(M(y), a_1 \cdots a_{i-1})$ for some string y of length k and some $a_1 \cdots a_{i-1} \in \{0,1\}^*$.

Lemma 2. *Let Γ be a class of languages and be closed under $\le_m^{\mathrm{P}}$-reductions. Then* NE $\subseteq \Gamma$ *if and only if* NEXP $\subseteq \Gamma$.

Proof: Since NE $\subseteq$ NEXP, it is trivial that NEXP $\subseteq \Gamma$ implies NE $\subseteq \Gamma$. We only prove that NE $\subseteq \Gamma$ implies NEXP $\subseteq \Gamma$. Assume NE $\subseteq \Gamma$. Let L be an arbitrary language in NEXP. Assume that $L \in \mathrm{NTIME}(2^{n^c})$ for some integer constant $c > 1$. Let $L' = \{x10^{|x|^c-|x|-1} : x \in L\}$. Since $L \in \mathrm{NTIME}(2^{n^c})$ with the constant c, we have $L' \in$ NE. We have a $\le_m^{\mathrm{P}}$-reduction $f(.)$ from L to L' with $f(x) = x10^{|x|^c-|x|-1}$ ($L \le_m^{\mathrm{P}} L'$). Since $L' \in$ NE $\subseteq \Gamma$ and Γ is closed under $\le_m^{\mathrm{P}}$-reductions, we have $L \in \Gamma$. Since L is an arbitrary language in NEXP, we have NEXP $\subseteq \Gamma$. ∎

Lemma 3. *Let $M_*(.)$ be a nondeterministic polynomial time oracle Turing machine. Let A be a language in* NP *and accepted by a polynomial time Turing machine $M_A(.)$. Then there is a nondeterministic $mn^{O(1)}$ time Turing machine $N(.)$ such that given the input $(m, M_*, M_A, 1^n)$,*

- *if $m \le |Q(M_*, A, n)|$, it outputs a subset of m different elements of $Q(M_*, A, n)$ in at least one path, and every path with nonempty output gives a subset of m different elements of $Q(M_*, A, n)$; and*

- *if $m > |Q(M_*, A, n)|$, it outputs empty set in every path.*

Proof: Let $M_A(.)$ be a polynomial time nondeterministic Turing machine that accepts A, and run in time n^{c_A} for a constant $c_A > 0$. Let $M_*(.)$ have time bound n^{c_*}. We design a nondeterministic Turing machine $N(.)$.

Let $N(.)$ do the following with input $(m, M_*, M_A, 1^n)$:

1. guess strings $x_1, \cdots, x_m$ of length n;
2. guess a path p_i and a series of oracle answers $a_{i,1} \cdots a_{i,j_i-1}$ for $M_*(x_i)$ for $i = 1, \cdots, m$;
3. if $M_*(x_i)$ makes the j_i-th query z_i on path p_i assuming the first the $j_i - 1$ oracle answers are $a_{i,1} \cdots a_{i,j_i-1}$;
4. then guess a path q_i for $M_A(z_i)$
5. if $z_1, \cdots, z_m$ are all different, and each z_i is accepted by $M_A(z_i)$ on path q_i
6. then output $z_1, \cdots, z_m$
7. else output the empty set $\emptyset$.

We note that line 3 is to check if z_i is in $H(M_*(x_i), a_{i,1} \cdots a_{i,j_i-1})$. Since $M_*(.)$ runs in time n^{c_*}, each z_i is of length at most n^{c_*}. The Turing machines $M_A(z_i)$ takes $|z_i|^{c_A} \leq n^{c_* c_A}$ time to accept z_i for $i = 1, \cdots, m$. Therefore, the total time of $N(.)$ with input $(m, M_*, M_A, 1^n)$ is $mn^{O(1)}$.

∎

Lemma 4. *Assume that S is in* NP *and S is nonexponentially dense. Then there is a $2^{n^{O(1)}}$ time computable nondecreasing function $e(1^n) : N \to N$ such that*

1. $|S^{\leq n}| \leq 2^{n^{\frac{1}{e(1^n)^2}}}$ *for infinitely many integers n;*
2. $e(1^{n^2}) \leq 2e(1^n)$ *for all n; and*
3. $\lim_{n\to\infty} e(1^n) = \infty$.

Proof: Let $e(1^0) = 1$. We construct $e(1^n)$ at phase n. Assume that we have constructed $e(1^1), \cdots, e(1^{t-1})$. Phase t below is for computing $e(1^t)$.

Phase t

1). Let k be the largest number less than t with $e(1^{k-1}) < e(1^k)$.

2). If $t \leq k^2$, then let $e(1^t) = e(1^k)$, and enter Phase $t + 1$.

3). If $t \neq j^{(e(1^k)+1)^2}$ for any integer j, then let $e(1^t) = e(1^k)$, and enter Phase $t + 1$.

4). Compute $s = |S^{\leq t}|$.

5). If $s \leq 2^{t^{\frac{1}{(e(1^k)+1)^2}}}$, then let $e(1^t) = e(1^k) + 1$.

End of Phase t.

The purpose of line 3 is to let $t = j^{(e(1^k)+1)^2}$ for some integer j after this line. This makes $t^{\frac{1}{(e(1^k)+1)^2}}$ be an integer and makes the computation easy at line 5. Checking the condition of the if statement at line 3 takes $t^{O(1)}$ time via a binary search. Computing s at step 4 in Phase t takes $2^{t^{O(1)}}$ steps since $S \in$ NP. Thus, function $e(1^n)$ is computable in $2^{n^{O(1)}}$ time. Since S is nonexponentially dense, the if condition in step 5 can be eventually satisfied and we have that $e(1^n)$ is unbounded.

Step 5 in Phase t makes function $e(.)$ satisfy condition 1 in the lemma. Step 2 and Step 5 in Phase t makes function $e(.)$ satisfy condition 2 in the lemma. The construction shows that $e(1^n)$ is nondecreasing since $e(1^t) \leq e(1^{t+1})$ for all integers t.

∎

Lemma 5. *Assume that $t(1^n)$ is nondecreasing unbounded function and $t(1^n)$ is computable in $2^{n^{O(1)}}$ time. Then there is a language $L_0 \in \mathrm{DTIME}(2^{n^{t(n)}})$ such that for every deterministic Turing machine $M(.)$ in time $2^{n^{O(1)}}$, $L(M)^{=n} \neq L_0^{=n}$ for all sufficiently large n.*

Proof: Let $M_1, \cdots, M_k, \cdots$ be the list of all deterministic Turing machines that each M_k runs in at most $2^{n^{t(1^n)/3}}$ time for all large n. The construction has infinitely phases for $n = 1, 2, \cdots$. It is easy to see that for each $2^{n^{O(1)}}$ time Turing machine $N(.)$, there is a $2^{n^{t(1^n)/3}}$ time Turing machine $M_i(.)$ with $L(M_i)^{=n} = L(N)^{=n}$ for all large n.

Phase n:

Let $x_1, \cdots, x_n$ be the first n $0,1$-strings of length n by the lexicographic order

For $i = 1, \cdots, n$, put x_i into $L_0^{=n}$ if and only if $L(M_i)(x_i)$ rejects.

End of Phase n.

According to the construction of phase n. The language L_0 can be computed in deterministic time $n \cdot 2^n \cdot 2^{n^{t(n)/3}} < 2^{n^{t(n)/2}}$ for all large n. By the construction of L_0, for each Turing machine M_i that runs in time $2^{n^{t(1^n)/3}}$, $L(M_i)^{=n} \neq L_0^{=n}$ for all large n.

∎

Theorem 6 and Theorem 7 are basically equivalent. They are the main separation results achieved in this paper. We will find more concrete complexity classes inside NP ∩ Nonexponentially-Dense-Class in section 5.

Theorem 6. $\mathrm{NEXP} \not\subseteq \mathrm{NP_T}(\mathrm{NP} \cap \text{Nonexponentially-Dense-Class})$.

Proof: Assume $\mathrm{NEXP} \subseteq \mathrm{NP_T}(\mathrm{NP} \cap \text{Nonexponentially-Dense-Class})$. We will bring a contradiction from this assumption. Since NEXP has a complete language K under $\leq_m^{\mathrm{P}}$ reductions, if $K \in \mathrm{NP_T}(S)$, then $\mathrm{NEXP} \subseteq \mathrm{NP_T}(S)$. Let S be a language in NP ∩ Nonexponentially-Dense-Class such that

$$\mathrm{NEXP} \subseteq \mathrm{NP_T}(S). \tag{1}$$

By Lemma 4, we have a nondecreasing unbounded function $e(1^n)$ that satisfies

$$e(1^{n^2}) \leq 2e(1^n) \tag{2}$$

and $(|S^{\leq n}|) \leq 2^{n^{1/e(1^n)^2}}$ for infinitely many integers n. Furthermore, function $e(1^n)$ is computable in $2^{n^{O(1)}}$ time. Let

$$h(n) = n^{e(1^n)}. \tag{3}$$

We apply the translational method to it. Let L_0 be an arbitrary language in $\mathrm{DTIME}(2^{h(n)})$, and accepted by a deterministic Turing machine $N(.)$ in $\mathrm{DTIME}(2^{h(n)})$ time. Define $L_1 = \{x10^{h(|x|)-|x|-1} : x \in L_0\}$.

Since function $e(1^n)$ is computable in $2^{n^{O(1)}}$ time, it is easy to see that L_1 is in $\mathrm{EXP} \subseteq \mathrm{NEXP}$. By our assumption (1), there is a nondeterministic polynomial time oracle Turing machine $M_1(.)$ for $L_1 \in \mathrm{NP_T}(S)$ (In other words, $M_1^S(.)$ accepts L). Assume that $M_1(.)$ runs in time n^{c_1} for all $n \geq 2$. Let $2 \leq u_1 < u_2 < \cdots < u_k < \cdots$ be the infinite list of integers such that

$$d_S(u_i) = |S|^{\leq u_i} \leq 2^{u_i^{1/e(1^{u_i})^2}}. \tag{4}$$

Define the language $L_2 = \{1^n 0 m : m \leq 2^n$ and there are at least m different strings $z_1, \cdots, z_m$ in $Q(M_1, S, h(n))\}$. Let n_i be the largest integers at least 2 such that

$$h(n_i)^{c_1} \leq u_i \tag{5}$$

for all large integers $i \geq i_0$ (it is easy to see the existence of such an integer i_0). Thus, we have

$$h(n_i + 1)^{c_1} > u_i. \tag{6}$$

For all large integers i, we have

$$e(1^{n_i}) \geq 8c_1 \tag{7}$$

since $e(1^n)$ is nondecreasing and unbounded. Since S is of density bounded by $d_S(n)$, the number of strings in S queried by $M_1(.)^S$ with inputs of length $h(n)$ is at most $d_S(h(n)^{c_1})$. In other words, we have

$$|Q(M_1, S, h(n))| \leq d_S(h(n)^{c_1}). \tag{8}$$

For the case $n = n_i$, we have the inequalities:

$$\begin{aligned}
d_S(h(n_i)^{c_1}) &\leq d_S(u_i) \qquad \text{(by inequality (5))} && (9)\\
&\leq 2^{u_i^{1/e(1^{u_i})^2}} \qquad \text{(by inequality (4))} && (10)\\
&\leq 2^{(h(n_i+1)^{c_1})^{1/e(1^{u_i})^2}} \qquad \text{(by inequality (6))} && (11)\\
&\leq 2^{h(n_i^2)^{c_1/e(1^{u_i})^2}} \qquad \text{(by the condition } n_i \geq 2) && (12)\\
&\leq 2^{(n_i^{2e(1^{n_i^2})})^{c_1/e(1^{n_i})^2}} \qquad \text{(by equation (3))} && (13)\\
&\leq 2^{(n_i^{4e(1^{n_i})})^{c_1/e(1^{n_i})^2}} \qquad \text{(by inequality (2))} && (14)\\
&< 2^{n_i} \qquad \text{(by inequality (7))} && (15)
\end{aligned}$$

By inequalities (9) to (15), and (8), we have the inequality

$$|Q(M_1, S, h(n_i))| < 2^{n_i} \qquad \text{for all large } i. \tag{16}$$

By Lemma 3, L_2 is in NEXP. By our assumption (1), $L_2 \in \mathrm{NP_T}(S)$ via some nondeterministic polynomial time oracle Turing machine $M_2(.)$. Assume that $M_2(.)$ runs in time n^{c_2} for all $n \geq 2$, where c_2 is a positive constant.

Define the language $L_3 = \{(x, m) : m \leq 2^{|x|}$ and there are at least m different strings $z_1, \cdots, z_m$ in $Q(M_1, S, h(n))$, and $M_1(x10^{h(|x|)-|x|-1})$ has an accept path that receives answer 1 for each query (to oracle S) in $\{z_1, \cdots, z_m\}$, and answer 0 for each query (to oracle S) not in $\{z_1, \cdots, z_m\}$ $\}$.

By Lemma 3, we have $L_3 \in$ NE. Thus, $L_3 \in \mathrm{NP_T}(S)$ via another nondeterministic polynomial time oracle Turing machine $M_3(.)$. Assume that $M_3(.)$ runs in time n^{c_3} for all $n \geq 2$.

In order to find the largest number m such that $1^n0m \in L_2$, m is always at most 2^n. Thus, the length of m is at most $n+1$. Using the binary search, we can find the largest m_{n_i} with $1^{n_i}0m_{n_i} \in L_2$ for $i = 1, 2, \cdots$. Let m_{n_i} be the largest m with $1^{n_i}0m \in L_2$ for $i = 1, 2, \cdots$. Since $S \in$ NP, m_{n_i} can be computed in $2^{{n_i}^{c_4}}$ time for some positive constant c_4 for all $i = 1, 2, \cdots$. By inequalityies (16), we have $m_{n_i} < 2^{n_i}$.

Claim 1. For $|x| = n_i$, we have $x10^{h(n_i)-n_i-1} \in L_1$ if and only if $(x, m_{n_i}) \in L_3$.

Proof: Assume that $z_1, \cdots, z_{m_{n_i}}$ are different elements in $Q(M_1, S, h(n))$. By the definition of m_{n_i}, a query if $y \in S$ made by $M_1^S(x10^{h(n_i)-n_i-1})$ to the oracle S is identical to checking if $y \in \{z_1, \cdots, z_{m_{n_i}}\}$. This is because all the strings in S that are queried are in the list $z_1, \cdots, z_{m_{n_i}}$. Thus, $x10^{h(n_i)-n_i-1} \in L_1$ if and only if $(x, m_{n_i}) \in L_3$. ∎

Assume that m_{n_i} is known. We just check if $(x, m_{n_i}) \in L_3$ with $|x| = n_i$. For $|x| = n_i$, we have $x \in L_0$ if and only if $x10^{h(n_i)-n_i-1} \in L_1$ if and only if $(x, m_{n_i}) \in L_3$ by Claim 1. Since $L_3 \in \mathrm{NP_T}(S)$ and $S \in$ NP, we only need $2^{n^{c_5}}$ time to decide if $(x, m_n) \in L_3$ for $n = n_1, n_2, \cdots$, where c_5 is a positive constant. Therefore, we can decide if $x \in L_0$ in $2^{{n_i}^{c_5}}$ time for $|x| = n_i$. Therefore, there is a deterministic Turing machine M_* that runs in $2^{n_i^{c_5}}$ time and has $L(M_*)^{=n_i} = L_0^{=n_i}$ for all i sufficiently large. Since L_0 is an arbitrary language in $\mathrm{DTIME}(2^{h(n)})$. Function $h(n)$ is a super-polynomial function. This brings there is a deterministic Turing machine M_* that runs in $2^{n^{c_5}}$ time and has $L(M_*)^{=n_i} = L_0^{=n_i}$ for all sufficiently large i, which contradicts Lemma 5.

∎

Theorem 7. NE $\not\subseteq \mathrm{NP_T}$(NP $\cap$ Nonexponentially-Dense-Class).

Proof: It follows from Lemma 2 and Theorem 6. ∎

Corollary 8. NEXP $\not\subseteq$ $\mathrm{NP_T}$(NP $\cap$ SPARSE).

Although it is hard to achieve NEXP $\neq$ $\mathrm{P_T}$(NP) or NEXP $\not\subseteq$ $\mathrm{P_T}$(SPARSE), we still have the following separation.

Corollary 9. NEXP $\not\subseteq$ $\mathrm{P_T}$(NP $\cap$ SPARSE).

5. Hard Low Density Problems in NP

It is natural to ask if there exists any hard low density problem in the class NP. In this section, we show the existence of low density sets in class NP. They are constructed from all natural NP-hard problems under the well known exponential time hypothesis that NP $\not\subseteq$ DTIME$(2^{n^{o(1)}})$ [14].

Definition 10.

- A function $g(n) : N \to N$ is *super-polynomial* if for every constant $c > 0$, $g(n) \geq n^c$ for all large n.
- A function $f(n) : N \to N$ is *sub-polynomial* if for every constant $c > 0$, $f(n) \leq n^c$ for all large n.
- A function $g(n) : N \to N$ is called well-super-polynomial if $g(n)$ is super-polynomial, $g(n)$ is time constructible, and there is a time constructible sub-polynomial function $f(n)$ such that $f(g(n)) \geq n$ for all sufficiently large n.
- A function $f(n) : N \to N$ is called well sub-polynomial if $f(n)$ is sup-polynomial, $f(n)$ is time constructible, and there is another time constructible super-polynomial function $h(n)$ such that for each positive constant c, $f(h(n)^c) \leq n$ for all sufficient large n.

Define $\log^{(1)} n = \log n = \lceil \log_2 n \rceil$. For integer $k \geq 1$, define $\log^{(k+1)} n = \log(\log^{(k)} n)$.

We provide the following lemma to give some concrete slowly growing well-sub-polynomial and well-super-polynomial functions.

Lemma 11.

1. *For each constant integer $k > 1$ and constant integer $a \geq 1$, the function $\left\lceil n^{1/(\log^{(k)} n)^a} \right\rceil$ is time constructible function from $N \to N$.*
2. *For each constant integer $k > 1$ and constant integer $a \geq 1$, the function $n^{(\log^{(k)} n)^a}$ is time constructible function from $N \to N$.*
3. *Assume k and a are fixed integers with $k > 1$ and $a > 1$. Let $f(n) = \left\lceil n^{1/(\log^{(k)} n)^a} \right\rceil$ and $h(n) = n^{(\log^{(k)} n)^{a-1}}$, then $f(h(n)) < n^{o(1)}$ for all large n.*
4. *Assume k and a are fixed integers with $k \geq 1$ and $a \geq 1$. Let $f(n) = \left\lceil n^{1/(\log^{(k)} n)^a} \right\rceil$ and $g(n) = n^{(\log^{(k)} n)^{a+1}}$, then $f(g(n)) > n$ for all large n.*

Proof: Statement 1: It takes $O(\log n)$ time to compute $\log^{(k)} n$. It takes another $O(\log n)$ time to compute $(\log^{(k)} n)^a$ since a is a constant. It takes another $O(\log n)$ time to compute $\left\lceil n^{1/(\log^{(k)} n)^a} \right\rceil$ via binary search. Since $\log n = o(\left\lceil n^{1/(\log^{(k)} n)^a} \right\rceil)$, we have that the function $\left\lceil n^{1/(\log^{(k)} n)^a} \right\rceil$ is time constructible.

Statement 2: It takes $O(\log n)$ time to compute $\log^{(k)} n$. It takes another $O(\log n)$ time to compute $m = (\log^{(k)} n)^a$ since a is a constant. Using the elementary method for multiplication, we can compute n^m with $O(m(\log n^m)^2) = O(m^2 \log n) = o(n^{(\log^{(k)} n)^a})$ time. Therefore, $n^{(\log^{(k)} n)^a}$ is time constructible.

Statement 3: We have

$$\begin{aligned} f(h(n)) &= f(n^{(\log^{(k)} n)^{a-1}}) \\ &= n^{O(\frac{1}{\log^{(k)} n})} \\ &< n \quad \text{for all large } n. \end{aligned}$$

Statement 4: We have

$$\begin{aligned} f(g(n)) &= f(n^{(\log^{(k)} n)^{a+1}}) \\ &= n^{\Omega(\log^{(k)} n)} \\ &> n \quad \text{for all large } n. \end{aligned}$$

▮

Definition 12.

- For a language A, let padding$(A, g(n))$ is the languages $L = \{x10^{g(|x|)-|x|-1} : x \in A\}$.
- For a class Λ of languages, define Padding$(\Lambda, g(n))$ to be the class of languages padding$(A, g(n))$ for all $A \in \Lambda$.

For example, let $g(n) = n^{(\log\log n)^k}$ for a fixed integer $k > 1$ and let $f(n) = n^{\frac{1}{(\log\log n)^{k-1}}}$. We have $2^{f(g(n))}) \geq 2^n$ for all sufficient large n.

Definition 13. A language A is of subexponential density if for each constant $c > 0$, $|A^{\leq n}| \leq 2^{n^c}$ for all large n.

Lemma 14. *Assume that A is a language and $g(n)$ is a super-polynomial function. Then* padding$(A, g(n))$ *is language of subexponential density.*

Proof: For each language A, there are at most 2^n strings of length n in A. When s is mapped into $s10^{g(|s|)-|s|-1)}$, its length becomes $g(|s|)$. Let c be an arbitrary positive constant. As $g(n)$ is a super-polynomial function, there is a constant integer $n_c \geq 2$ such that for every $n > n_c$,

$$g(n) > n^{10/c}. \tag{17}$$

Let $m_c = n_c^{\frac{10}{c}}$. We have $n_c = m_c^{\frac{c}{10}}$. Let m be an arbitrary number greater than m_c. Let k be the largest integer with $g(k) \leq m$.

Case 1: $k \leq n_c$. The number of strings of length at most k is at most $2 \cdot 2^k \leq 2^{2k} \leq 2^{2n_c} < 2^{n_c^2} \leq 2^{m_c^{\frac{c}{5}}} < 2^{m^{\frac{c}{5}}}$. Therefore, the number of strings of length at most m in padding$(A, g(n))$ is at most $2^{m^{\frac{c}{5}}}$.

Case 2: $k > n_c$. We have $m \geq g(k) > k^{\frac{10}{c}}$ by inequality (17). Thus, $k < m^{\frac{c}{10}}$. The number of strings of length at most k at is no more than $2 \cdot 2^k < 2^{2k} < 2^{k^2} < 2^{m^{\frac{c}{5}}}$. Therefore, the number of strings of length at most m in padding$(A, g(n))$ is at most $2^{m^{\frac{c}{5}}}$.

In every case, we have $|\text{padding}(A, g(n))^{\leq m}| \leq 2^{m^{\frac{c}{5}}}$. Since c is an arbitrary positive constant, padding$(A, g(n))$ is a language of subexponenital density by Definition 13. ▮

Lemma 15. *Assume that A is a language and $g(n)$ is a strictly increasing super-polynomial function and $f(n)$ is a sub-polynomial function with $f(g(n)) \geq n$, then* padding$(A, g(n))$ *is language of density $O(2^{f(n)})$.*

Proof: For each language A, there are at most 2^n strings of length n in A. When s is mapped into $s10^{g(|s|)-|s|-1)}$, its length becomes $g(|s|)$. Since $g(n)$ is increasing super-polynomial function, $g(n) < g(n+1)$ for all large n. We have $2^n \leq 2^{f(g(n))}$. Thus, padding$(A, g(n))$ is language of density $O(2^{f(n)})$. ▮

We have Theorem 16 that shows the existence of subexponential density sets that are still far from polynomial time computable under the reasonable assumption that NP $\not\subseteq$ DTIME$(2^{n^{o(1)}})$.

Theorem 16. *Assume that $g(n)$ is a strictly increasing well-super-polynomial function, and $f(n)$ is a sub-polynomial function with $f(g(n)) \geq n$. If $\mathrm{NP} \not\subseteq \mathrm{DTIME}(2^{n^{o(1)}})$, then for every NP-complete language A, $\mathrm{padding}(A, g(n))$ is a language of density of $\mathrm{Density}(2^{f(n)})$, and not in $\mathrm{DTIME}(T(n))$, where $T(n)$ is an arbitrary function with $T(g(n)) = 2^{n^{o(1)}}$.*

Proof: Let A be a NP-complete language. The density of $\mathrm{padding}(C, g(n))$ follows from Lemma 15. If $\mathrm{padding}(C, g(n))$ is computable in time $T(n)$, we have that A is computable in time $T(g(n)) = 2^{n^{o(1)}}$. Thus, $\mathrm{NP} \subseteq \mathrm{DTIME}(2^{n^{o(1)}})$. This contradicts the condition $\mathrm{NP} \not\subseteq \mathrm{DTIME}(2^{n^{o(1)}})$. ∎

The following corollary gives concrete result by assigning concrete functions for $f(n), g(n)$ and $T(n)$.

Corollary 17. *Let $g(n) = n^{(\log^{(k)})^a}$, $f(n) = \left\lceil n^{\frac{1}{(\log^{(k)} n)^{a-1}}} \right\rceil$, and $T(n) = 2^{\left\lceil n^{1/(\log^{(k)} n)^{a+1}} \right\rceil}$ with fixed integers $a > 1$ and $k > 1$. If $\mathrm{NP} \not\subseteq \mathrm{DTIME}(2^{n^{o(1)}})$, then for every NP-complete language A, $\mathrm{padding}(A, g(n))$ is a language of density of $\mathrm{Density}(2^{f(n)})$, and not in $\mathrm{DTIME}(T(n))$.*

Proof: For $g(n) = n^{(\log^{(k)} n)^a}$ and $f(n) = n^{\frac{1}{(\log^{(k)} n)^{a-1}}}$. By statement 4 of Lemma 11, we have $f(g(n)) \geq n$.

For $T(n) = 2^{\left\lceil n^{1/(\log^{(k)} n)^{a+1}} \right\rceil}$ for an arbitrary constant $c > 0$. By statement 3 of Lemma 11, we have $T(g(n)) = 2^{n^{o(1)}}$. The three functions satisfy the conditions in Theorem 16. The corollary follows from Theorem 16. ∎

We separate both NEXP and NE from $\mathrm{NP_T}(\mathrm{padding}(\mathrm{NP}, g(n)))$ for any super-polynomial time constructible function $g(n)$ from N to N in Theorems 18 and 19. For a given $g(n) : N \to N$, $\mathrm{NP_T}(\mathrm{padding}(\mathrm{NP}, g(n)))$ is a concrete computational complexity class.

Theorem 18. *Assume that $g(n)$ is a super-polynomial time constructible function from N to N. Then $\mathrm{NEXP} \not\subseteq \mathrm{NP_T}(\mathrm{padding}(\mathrm{NP}, g(n)))$.*

Proof: It follows from Lemma 14 and Theorem 6. ∎

Theorem 19. *Assume that $g(n)$ is a super-polynomial time constructible function from N to N. Then $\mathrm{NE} \not\subseteq \mathrm{NP_T}(\mathrm{padding}(\mathrm{NP}, g(n)))$.*

Proof: It follows from Lemma 2 and Theorem 18. ∎

6. Separating NEXP from $\mathrm{NP_T}(\mathrm{P}_{tt}(\mathrm{NP}) \cap \mathrm{TALLY})$

In this section, we separate NEXP from $\mathrm{NP_T}(\mathrm{P}_{tt}(\mathrm{NP}) \cap \mathrm{TALLY})$. A more generalized theorem is given by Theorem 24. We are more carefully to combine the counting method with the translational method to prove it.

Definition 20.

- Let M_1 be a nondeterministic oracle Turing machine and M_2 be a deterministic oracle Turing machine. Define $M_1^{M_2}$ be a nondeterministic Turing machine such that $M_1(x)$ takes an input x, each query y produced by M_1 is answered by $M_2(y)$, which will access an oracle during the computation.
- Let M_1 be a nondeterministic oracle Turing machine and M_2 be a deterministic oracle Turing machine. Let A be an oracle for M_2. Define $(M_1^{M_2})^A$ be a nondeterministic Turing machine $M_1^{M_2}$ with oracle A such that $M_1(x)$ takes an input x, each query y produced by M_1 is answered by $M_2^A(y)$.

Definition 21.

- For an oracle Turing machine M and an integer k, define $PQ(M,y,k)$ to be the union of all $H(M(y), a_1\cdots a_{i-1})$ (see Definition 1) with $i\le k$ and $a_1\cdots a_{i-1}\in\{0,1\}^{\le k}$.
- Assume that M_1 is a nondeterministic Turing machine and M_2 is a deterministic adaptive oracle Turing machine $M(.)$. Let A be an oracle set, and k is an integer. Define

$$Q_1(M_1^{M_2}, A, B, k_1, k_2, m) = \bigcup_{z\in\left(\bigcup_{y\in B^{=m}} PQ(M_1,y,k_1)\right)} (A\cap PQ(M_2, z, k_2)). \tag{18}$$

The purpose of Lemma 22 for the proof of Theorem 24 is similar to Lemma 3 for Theorem 6.

Lemma 22. *Assume that $A\in$ NP, $B\in$ NP and $M_1(.)$ and $M_2()$ are polynomial time nondeterministic Turing machines. Then there is a nondeterministic machine $N(.)$ such that given the input $(M_1^{M_2}, M_A, M_B, k_1, k_2, 1^n)$, if $m\le |Q_1(M_1^{M_2}, A, B, k_1, k_2, n)|$, it outputs a subset of m different elements of $Q_1(M_1^{M_2}, A, B, k_1, k_2, n)$ in time $mn^{O(1)}$ in at least one path; and otherwise, it outputs empty set in every path, where M_A is an polynomial time nondeterministic Turing machine to accept A, and M_B is a polynomial time nondeterministic Turing machine to accept B.*

Proof: We design a nondeterministic Turing machine $N(.)$. Let $N(.)$ do the following with input $(M_*, M_A, M_B, k_1, k_2, 1^n)$:

1. guess strings $x_1,\cdots,x_m$ of length n,
2. guess a path h_i of $M_B(x_i)$ for each x_i,
3. guess a path p_i and a query y_i for each $M_1(x_i)$,
4. guess a path w_i and a query z_i for each $M_2(y_i)$, and
5. guess a path q_i for $M_A(z_i)$ for $i=1,\cdots,m$.
6. If $M_B(x_i)$ accepts in path h_i, $M_1(x_i)$ queries y_i in path p_i for $i=1,\cdots,m$, $M_2(y_i)$ queries z_i in path w_i for $i=1,\cdots,m$, and $M_A(z_i)$ accepts in path q_i for $i=1,\cdots,m$, then N outputs all $z_1,\cdots,z_m$. Otherwise, N outputs $\emptyset$.

Note that for a path p_i and a query y_i for $M_1(x_i)$, a part of path p_i is $a_1\cdots a_{j-1}, j$ with $j\le k_1$ such that $M_1(x_i)$ follows path p_i and its j-th query is y_i assuming it receives the $j-1$ answers are $a_1\cdots a_{j-1}$.

Note that for a path w_i and a query z_i for $M_2(y_i)$, a part of path w_i is $b_1\cdots b_{j-1}, j$ with $j\le k_2$ such that $M_2(y_i)$ follows a path w_i and its j-th query is z_i assuming it receives the $j-1$ answers are $b_1\cdots b_{j-1}$.

Since $M_1(.), M_2(.), M_A(.)$ and $M_B(.)$ all run in polynomial time, we have that the time for $N(.)$ is bounded by $mn^{O(1)}$. ∎

Definition 23. For a set B, define $\wp(B)$ to be the *power set* of B (the class of all subsets of B).

Theorem 24 gives another separation for NEXP from the polynomial time hierarchy. It is incomparable with Theorem 6.

Theorem 24. *Assume that B is an language in* (NP ∩ co-NP) ∩ Nonexponentially-Dense-Class. *Then for any well sub-polynomial function $g(n)$,* $\mathrm{NEXP}\not\subseteq \mathrm{NP_T}(\mathrm{P}_{g(n)-\mathrm{T}}(\mathrm{NP})\cap\wp(B))$.

Proof: We use a combination of counting method and translational method to prove this theorem. Let M_B be a polynomial time nondeterministic Turing machine to accept B, and $M_{\overline{B}}$ be a polynomial time nondeterministic Turing machine to accept $\overline{B}$. Let SAT be the well known NP-complete problem.

Assume $\mathrm{NEXP}\subseteq \mathrm{NP_T}(\mathrm{P}_{g(n)-\mathrm{T}}(\mathrm{NP})\cap\wp(B))$. Since NEXP has a complete language under $\le_m^{\mathrm{P}}$-reductions, we assume $\mathrm{NEXP}\subseteq\mathrm{NP_T}(K)$ for some $K\subseteq B$ and also $K\in\mathrm{P}_{g(n)-\mathrm{T}}(\mathrm{NP})$. Let $K\in \mathrm{P}_{g(n)-\mathrm{T}}(\mathrm{SAT})$ via oracle

Turing machine $M_q(.)$. Let n^{c_q} be the running time of M_q. Since $B \in \mathrm{NP} \cap \text{co-NP}$ and is of nonexponential density, by Lemma 4, we have $e(1^n)$ to be a nondecreasing function with $\lim_{n\to\infty} e(1^n) = \infty$,

$$e(1^{n^2}) \le 2e(1^n), \tag{19}$$

and $(|B^{\le n}|) \le 2^{n^{1/e(1^n)^2}}$ for infinitely many integers n. Furthermore, function $e(1^n)$ is computable in $2^{n^{O(1)}}$ time.

Since $g(n)$ is a well sub-polynomial function, let $h_g(n)$ be a well super-polynomial function (see Definition 10) such that for each positive constant c,

$$g(h_g(n)^c) \le n \qquad \text{for all large integers } n. \tag{20}$$

Let

$$h(n) = \min(n^{e(1^n)}, h_g(n), 2^n). \tag{21}$$

We apply the translational method to it. Let L be an arbitrary language in $\mathrm{DTIME}(2^{h(n)})$. Define $L_1 = \{x10^{h(n)} : x \in L\}$.

Since $e(1^n)$ is computable in $2^{n^{O(1)}}$ time and $h_g(n)$ is time constructible, we have that L_1 is in NEXP. Let $L_1 \in \mathrm{NP_T}(K)$ via an nondeterministic oracle Turing machine $M_1(.)$ with oracle K. Assume that $M_1(.)$ runs in time n^{c_1} for all $n \ge 2$. Let $u_1 < u_2 < \cdots < u_k < \cdots$ be the infinite list of integers at least 2 such that

$$d_B(u_i) = (|B|^{\le u_i}) \le 2^{u_i^{1/e(1^{u_i})^2}}. \tag{22}$$

Let n_i be the largest integers at least 2 such that

$$h(n_i)^{c_1} \le u_i \tag{23}$$

for all sufficiently large integers i. Thus, we have

$$h(n_i + 1)^{c_1} > u_i. \tag{24}$$

Define $L_2 = \{1^n0m : m \le 2^{2n}$ there are m different elements in $Q_1(M_1^{M_q}, \mathrm{SAT}, B, h(n)^{c_1}, g(h(n)^{c_1}), h(n))$ $\}$.

The number of strings $z \in B$ queried by $M_1(.)$ with inputs of length $h(n)$ is at most $d_B(h(n)^{c_1})$ since the length of z is at most $h(n)^{c_1}$. In other words,

$$\left| \bigcup_{y \in \Sigma^{=h(n)}} PQ(M_1, y, h(n)^{c_1}) \right| \le d_B(h(n)^{c_1}). \tag{25}$$

As M_q is a deterministic oracle Turing machine with the number of queries bounded by function $g(.)$, we have

$$|PQ(M_q, z, g(h(n)^{c_1})| \le 2^{g(h(n)^{c_1})} \qquad \text{for each } z \text{ of length at most } h(n)^{c_1}. \tag{26}$$

By equations (18), (25), and (26), we have the inequality

$$|Q_1(M_1^{M_q}, \mathrm{SAT}, B, h(n)^{c_1}, g(h(n)^{c_1}), h(n))| \le d_B(h(n)^{c_1}) 2^{g(h(n)^{c_1})} \qquad \text{for all large } n. \tag{27}$$

We will show this number is less than 2^{2n_i} if $n = n_i$ for all large i. For all large i, we have

$$e(1^{n_i}) \ge 8c_1 \tag{28}$$

since $e(1^n)$ is nondecreasing and unbounded. Since B is of density bounded by $d_B(n)$, we have the inequalities

$$
\begin{aligned}
d_B(h(n_i)^{c_1}) &\leq d_B(u_i) \quad \text{(by inequality (23))} && (29)\\
&\leq 2^{u_i^{1/e(1^{u_i})^2}} \quad \text{(by inequality (22))} && (30)\\
&\leq 2^{(h(n_i+1)^{c_1})^{1/e(1^{u_i})^2}} \quad \text{(by inequality (24))} && (31)\\
&\leq 2^{h(n_i^2)^{c_1/e(1^{u_i})^2}} \quad \text{(by the condition } n_i \geq 2) && (32)\\
&\leq 2^{(n_i^{2e(1^{n_i^2})})^{c_1/e(1^{n_i})^2}} \quad \text{(by equation (21))} && (33)\\
&\leq 2^{(n_i^{4e(1^{n_i})})^{c_1/e(1^{n_i})^2}} \quad \text{(by equation (19))} && (34)\\
&< 2^{n_i}. \quad \text{(by inequality (28))} && (35)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
d_B(h(n_i)^{c_1})2^{g(h(n_i)^{c_1})} &\leq d_B(h(n_i)^{c_1}) \cdot 2^{n_i} \quad \text{(by inequality (20) and equation (21))} && (36)\\
&< 2^{2n_i}. \quad \text{(by inequality (35))} && (37)
\end{aligned}
$$

By inequalities (27), and (37)

$$|Q_1(M_1^{M_q}, \text{SAT}, B, h(n_i)^{c_1}, g(h(n_i)^{c_1}), h(n_i))| < 2^{2n_i} \qquad \text{for all large } i. \tag{38}$$

We can assume that $m \leq 2^{2n}$ (otherwise, $1^n 0m \notin L_2$). Since $M_1(.)$ and $M_q(.)$ run in n^{c_1} and n^{c_q} time, respectively, we have that $M_1^{M_q}(.)$ runs in $n^{c_1 c_q}$ time. By Lemma 22, the decision if $1^n 0m \in L_2$ can be made by a nondeterministic Turing machine in $mh(n)^{O(1)} = 2^{O(n)}$ time for all large n. We have $L_2 \in$ NEXP. Thus, $L_2 \in \text{NP}_\text{T}(K)$ via a nondeterministic Turing machine $M_2(.)$. Since $K \in \text{P}_T^{\text{NP}}$, there is a constant c_2 such that we can find the largest m_n in time $2^{n^{c_2}}$ such that $1^n 0m \in L_2$.

Define the language $L_3 = \{(x, m)$: there are at least m different strings $z_1, \cdots, z_m$ in $Q_1(M_1^{M_q}, \text{SAT}, B, h(n)^{c_1}, g(h(n)^{c_1}), h(n))$, and $(M_1^{M_q})^{\text{SAT}}(x10^{h(|x|)-|x|-1})$ has an accept path that receives answer 1 for each query (to SAT), which is generated by some $y \in B$, in $\{z_1, \cdots, z_m\}$, and answer 0 for each query (to SAT), which is generated by some $y \in B$, not in $\{z_1, \cdots, z_m\}$ $\}$.

By Lemma 3, we have $L_3 \in$ NE. Thus, $L_3 \in \text{NP}_\text{T}(K)$ via another polynomial time nondeterministic Turing machine $M_3(.)$. Assume that $M_3(.)$ runs in time n^{c_3} for all $n \geq 2$.

Assume $n_i = |x|$. In order to find the largest number m such that $1^{n_i} 0m \in L_2$, m is always at most 2^{2n_i}. Thus, the length of m is at most $2n$. Using the binary search, we can find the largest m with $1^{n_i} 0m \in L_2$. Let m_{n_i} be the largest m with $1^{n_i} 0m \in L_2$. Since SAT $\in$ NP, m_{n_i} can be computed in $2^{n_i^{c_4}}$ deterministic time for some positive constant c_4.

Assume that m_{n_i} is known. We just check if $(x, m_{n_i}) \in L_3$, where $n_i = |x|$. It is easy to see $x \in L$ if and only if $x10^{h(n_i)-n_i-1} \in L_1$ if and only if $(x, m_{n_i}) \in L_3$. Since $L_3 \in \text{NP}_\text{T}(K)$ and $K \in \text{P}_T^{\text{NP}}$, we only need $2^{n_i^{c_5}}$ time to decide if $(x, m_{n_i}) \in L_3$, where c_5 is a positive constant. Therefore, we can decide if $x \in L$ in $2^{n_i^{c_5}}$ time.

Therefore, there is a deterministic Turing machine $M_*(.)$ that runs in $2^{n_i^{c_5}}$ time and accepts L^{n_i} for all i sufficiently large. Note that L is an arbitrary language in DTIME$(2^{h(n)})$, and function $h(n)$ is a super-polynomial function. This brings there is a deterministic Turing machine $M_*(.)$ that runs in $2^{n^{c_5}}$ time and accepts L^{n_i} for all sufficiently large integers i. This contradicts Lemma 5. ∎

It is easy to see that $\{1\}^*$ is a sparse language in P $\subseteq$ NP $\cap$ co-NP, and TALLY is the power set of $\{1\}^*$. We have the following corollary.

Corollary 25. NEXP $\not\subseteq \text{NP}_\text{T}(\text{P}_{g(n)-\text{T}}(\text{NP}) \cap \text{TALLY})$ *for any well sub-polynomial function* $g(n)$.

It is well known that $\text{P}_{tt}(\text{NP}) = \text{P}_{O(\log n)-\text{T}}(\text{NP})$ (see [6]), we have corollary 26.

Corollary 26. NEXP $\not\subseteq \text{NP}_\text{T}(\text{P}_{tt}(\text{NP}) \cap \text{TALLY})$.

7. Conclusions

We show that NEXP $\not\subseteq$ $\mathrm{NP_T}$(NP $\cap$ Nonexponentially-Dense-Class). This result has almost reached the limit of relativizable technology. A fundamental open problem is to separate NEXP from BPP. We would like to see the further step toward this target. Our method is a relativizable. Since there exists an oracle to collapse NEXP to BPP by Heller [12], separating NEXP from BPP requires a new way to go through the barrier of relativization. We feel that it is easy to extend results to super polynomial time classes such as NE $\not\subseteq$ $\mathrm{NTIME}(n^{O(\log n)})_{\mathrm{T}}(\mathrm{NTIME}(n^{O(\log n)}) \cap \mathrm{SPARSE})$. We will present this kind of results in the extended version of this paper.

References

[1] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[2] E. Allender, R. Beigel, U. Hertrampf, and S. Homer. Almost-everywhere complexity hierarchies for nondeterministic time. *Theoretical Computer Science*, 115:225–241, Aug. 1993.

[3] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SICOMP*, 6:305–322, 1977.

[4] H. Buhrman, L. Fortnow, and R. Santhanam. Unconditional lower bounds against advice. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming, 36th International Colloquium (ICALP'09)*, pages 195–209, 2009.

[5] H. Buhrman and L. Torenvliet. On the cutting edge of relativization: The resource bounded injury method. In *Proceedings of the 21st International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science 820, Springer*, pages 263–273, 1994.

[6] S. R. Buss and L. E. Hay. On truth table reducibility to SAT. *Inf. & Comp.*, 91(1):86–102, Mar. 1991.

[7] S. Cook. A hierarchy for nondeterministic time complexity. *JCSS*, 7:343–353, 1973.

[8] B. Fu. With quasi-linear queries exp is not polynomial-time turning reducible to sparse sets. *SIAM Journal on Computing*, pages 1082–1090, 1995.

[9] B. Fu, A. Li, and L. Zhang. Separating ne from some nonuniform nondeterministic complexity classes. In *In Proceedings of the 15th Annual International Conference in Computing and Combinatorics, Lecture Notes in Computer Science 5609*, pages 486–495, 2009.

[10] B. Fu, H.-Z. Li, and Y. Zhong. An application of the translational method. *Mathematical Systems Theory*, 27:183–186, 1994.

[11] R. Harkins and J. Hitchcock. Dimension, halfspaces, and the density of hard sets. In *In Proceedings of the 13th Annual International Conference Computing and Combinatorics (COCOON 2007), Lecture Notes in Computer Science 4598, 2007*, pages 129–139, 2007.

[12] H. Heller. On relativized exponential and probabilistic complexity classes. *Inf. & Comp.*, 71:231–243, 1986.

[13] J. Hitchcock. Online learning and resource-bounded dimension: Winnow yields new lower bounds for hard sets. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS 2006), Lecture Notes in Computer Science 3884*, pages 408–419, 2006.

[14] R. Impagliazzo and R. Paturi. The complexity of k-sat. In *Proceedings of the 14th IEEE Conference on Computational Complexity*, page 237 240, 1999.

[15] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 302–309, 1980.

[16] J. Lutz and E. Mayordomo. Measure, stochasticity, and the density of hard languages. *SIAM Journal on Computing*, 23(4):762–779, 1994.

[17] S. Mahaney. Sparse complete sets for NP: Solution to a conjecture of Berman and Hartmanis. *JCSS*, 25:130–143, 1982.

[18] S. Mocas. Separating classes in the exponential-time hierarchy from classes in ph. *Theor. Comput. Sci.*, 158:221–231, 1996.

[19] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of np sets to sparse sets. *SIAM J. Comput.*, 20(3):471–483, 1991.

[20] J. Seiferas, M. J. Fischer, and A. Meyer. Separating nondeterministic time complexity classes. *Journal of ACM*, 25:146167, 1978.

[21] O. Watanabe. Polynomial time reducibility to a set of small density. In *Proceedings of the 2nd IEEE Structure in Complexity Theory Conference*, pages 138–146, 1987.

[22] R. Williams. Non-uniform acc circuit lower bounds. http://www.cs.cmu.edu/ ryanw/acc-lbs.pdf, 2010.

[23] S. Zak. A turing machine hierarchy. *Theoretical Computer Science*, 26:327333, 1983.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de