Partial Bi-immunity, Scaled Dimension, and NP-Completeness

John M. Hitchcock* A. Pavan† N. V. Vinodchandran‡

Abstract

The Turing and many-one completeness notions for NP have been previously separated under *measure*, *genericity*, and *bi-immunity* hypotheses on NP. The proofs of all these results rely on the existence of a language in NP with almost everywhere hardness.

In this paper we separate the same NP-completeness notions under a *partial bi-immunity* hypothesis that is weaker and only yields a language in NP that is hard to solve on most strings. This improves the results of Lutz and Mayordomo (*Theoretical Computer Science*, 1996), Ambos-Spies and Bentzien (*Journal of Computer and System Sciences*, 2000), and Pavan and Selman (*Information and Computation*, 2004). The proof of this theorem is a significant departure from previous work. We also use this theorem to separate the NP-completeness notions under a *scaled dimension* hypothesis on NP.

1 Introduction

Completeness is, arguably, the single most important concept in complexity theory. Many well-studied complexity classes have complete problems. Complete problems capture the inherent difficulty of a class. Informally, a language L is complete for a class if L belongs to the class and every language in the class is *polynomial-time reducible* to L. However, there is no unique notion of a polynomial-time reduction. Various types of reductions give rise to various notions of completeness. A comparison of these notions of completeness helps us understand the structure of a complexity class.

Cook [5], in his paper on NP-completeness, used Turing reductions, whereas Karp [14] and Levin [17], in their papers on NP-completeness, used many-one reductions. Informally, with Turing reductions an instance of a problem can be solved by asking polynomially many queries about the instances of another problem. Moreover, these queries can be adaptive, i.e, a query can depend on the answers to the previous queries. Many-one reductions are more restrictive. Here we require instances of one problem to be mapped into instances of the other. It is easy to see that if L is complete under many-one reductions, then L is complete under Turing reductions. The interesting question is whether the converse is true.

*Department of Computer Science, University of Wyoming, Laramie, WY 82071. jhitchco@cs.uwyo.edu. Part of this research was done while this author was visiting the University of Nebraska-Lincoln. Research supported in part by NSF grant CCF-0515313.

†Department of Computer Science, Iowa State University, Ames, IA 50011. pavan@cs.iastate.edu. Research supported in part by NSF grants CCR-0344187 and CCF-0430807.

‡Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588. vinod@cse.unl.edu. Research supported in part by NSF grant CCF-0430991 and University of Nebraska Layman Award.

It seems, to capture the intuition that if a complete problem is easy, then the entire class is easy, Turing reductions are "correct" reductions to define completeness. This leads us to the following question: for a class are there languages that are complete using Turing reductions, but not complete using many-one reductions? This is one of the outstanding questions in the field.

These questions are well studied for exponential-time classes such as EXP and NEXP. For example, Ladner, Lynch and Selman [16] showed that there exist languages A and B in E such that A is Turing reducible to B but not many-one reducible to B. Ko and Moore [15] demonstrated the existence of $\leq_T^p$-complete sets for EXP that are not $\leq_m^p$-complete. Watanabe [25] extended this result significantly, showing that $\leq_{1tt}^p$, $\leq_{btt}^p$, $\leq_{tt}^p$, and $\leq_T^p$-completeness are mutually different, while Homer, Kurtz, and Royer [13] proved that $\leq_m^p$ and $\leq_{1tt}^p$-completeness are identical. Buhrman, Homer, and Torenvliet [3] achieved the separation for NEXP. See the survey articles by Homer [11, 12] and Buhrman and Torenvliet [4] for more details.

However, the progress on understanding the behavior of reducibilities within NP has been painfully slow. One of the first results is due to Selman [24] who showed under a reasonable assumption that Turing reductions and many-one reductions differ in NP. Longpré and Young [18] showed that for every polynomial t, there exist a language L that is Turing complete for NP but not many-one complete under $t(n)$-time bounded reductions. However, L is many-one complete if we allow slower reductions. Thus the question of whether Turing completeness is different from many-one completeness for NP remained open for a long time.

Lutz and Mayordomo [20] were the first to separate these NP-completeness notions under a plausible hypothesis. They showed that under the *measure hypothesis*, which asserts that "NP does not have p-measure 0," there exists a 2-Turing complete language for NP that is not many-one complete. Ambos-Spies and Bentzien [1] extended this result significantly. They used the *genericity hypothesis* that asserts "NP has a p-generic language", which is weaker than the measure hypothesis, to separate 2-Turing completeness from many-one completeness. In addition, Ambos-Spies and Bentzien separate nearly all NP-completeness notions for bounded truth-table reducibilities. Pavan and Selman [23] showed that the *bi-immunity hypothesis*, which says NP contains a 2^{n^ϵ}-bi-immune language, implies 2-Turing completeness is different from many-one completeness for NP. Since the genericity hypothesis implies the bi-immunity hypothesis, this improved the results of Lutz and Mayordomo, and Ambos-Spies and Bentzien. Pavan [22] surveys these results.

However, all known hypotheses that separate completeness notions within NP, such as the measure hypothesis, the genericity hypothesis, and the bi-immunity hypothesis, assume the existence of a language L in NP with *almost everywhere hardness*, i.e., every machine that decides L takes more than subexponential time (2^{n^ϵ}) on *all but finitely many inputs*. Thus the next logical step is to relax the almost everywhere hardness condition. In particular, it is unknown whether we can separate completeness notions under any kind of *average-case* hardness assumptions. In this paper we make progress toward this direction.

We separate completeness notions under a certain (weak) average-case hardness assumption which we call the *partial bi-immunity* hypothesis. Informally, a language L is partial bi-immune if every machine that decides L takes at least 2^{n^ϵ} time on all but $2^{n^{o(1)}}$ strings of length n (a similar partial immunity notion is studied by Grollmann and Selman [6]). The partial bi-immunity hypothesis asserts that NP contains a language that is partial bi-immune. We show as our main theorem that under the partial bi-immunity hypothesis, Turing completeness is different from many-one completeness for NP. We note that the proofs of [20, 1, 23] do not go through if we replace their respective hypotheses with the partial bi-immunity hypothesis. It is essential that the language is almost everywhere hard for those proofs. On the other hand, those proofs do not really make use of properties of NP – the arguments go through for any class that is closed under union, intersection, and disjoint unions. We achieve our result by making crucial use that the partial bi-immune language is in NP and using very different techniques. In particular, we make use of the unique property of NP – polynomial-time verifiability – which allows us to define left sets, a useful tool introduced by Ogiwara and Watanabe [21]. Thus the partial bi-immunity hypothesis becomes the *weakest* known hypothesis that separates Turing completeness from many-one completeness for NP.

Lutz [19] recently introduced *resource-bounded dimension*, an extension of classical Hausdorff dimension that is useful for distinguishing among resource-bounded measure 0 classes. Dimension hypotheses such as "NP has positive p-dimension" or "NP has p-dimension 1," which are weaker than the measure hypothesis, can be studied for their consequences [7]. While the measure hypothesis asserts the existence of language in NP that is almost everywhere hard, a dimension hypothesis only implies the existence of a language in NP that is only *average-case hard* in some sense. Thus it is interesting to ask whether Turing completeness can be separated from many-one completeness using a dimension hypothesis. We give a partial answer to this question.

We show that if a certain *scaled dimension* [9] hypothesis holds, then Turing completeness for NP is different from many-one completeness. The scaled dimension hypothesis asserts that NP has positive scaled p-dimension in the $-3rd$ order. This scaled dimension hypothesis (which has also been used in [10]) is stronger than the dimension hypotheses, but weaker than the measure hypothesis. This hypothesis appears to be incomparable with the genericity and bi-immunity hypotheses. We use previous work on scaled dimension [8] and our main theorem to derive this result.

2 Preliminaries

A is *many-one reducible* to B, $A \leq_m^p B$, if there is a polynomial-time computable function f such that x belongs to A if and only if $f(x)$ is in B. A is *Turing reducible* to B, $A \leq_T^p B$, if A can be decided by a polynomial-time bounded oracle machine that is allowed to make membership queries to B. Note that the queries to B can be adaptive. We say A is *k-Turing reducible* to B if A is Turing reducible to B via a oracle machine that makes k queries. Given a reduction $\leq_r^p$, a set S is $\leq_r^p$-complete for NP if $S \in$ NP and every set in NP is $\leq_r^p$-reducible to S. We assume without explicitly mentioning that any fraction is rounded up to get an integer if required.

3 Partial Bi-immunity

In this section we show that the partial bi-immunity hypothesis implies the separation of Turing completeness from many-one completeness for NP.

First we review the notion of almost-everywhere hardness. An infinite language is *immune* to a complexity class $\mathcal{C}$ or is $\mathcal{C}$-immune, if no infinite subset of L belongs to $\mathcal{C}$. An infinite language is *$\mathcal{C}$-bi-immune* if both L and $\overline{L}$ are $\mathcal{C}$-immune. Balcázar and Schöning [2] observed that a language L is DTIME($t(n)$)-bi-immune if and only if every machine that correctly decides L takes more than $t(n)$ time on all but finitely many strings.

Bi-immunity Hypothesis: For some $\epsilon > 0$, NP contains a DTIME(2^{n^ϵ})-bi-immune language.

We now introduce the notion of *partial bi-immunity.*

Definition. A language L is *$(t(n), q(n))$-partial-bi-immune*, if for every machine M that correctly decides L, for all but finitely many n, M takes more than $t(n)$ time on all but $q(n)$ strings of length

n.

In other words, if a language L is $(t(n), q(n))$-partial-bi-immune, then for every machine M that decides L, for all but finitely many n

$$|\{x \in \Sigma^n \mid T_M(x) \leq t(n)\}| \leq q(n),$$

where $T_M(x)$ denotes the running time of M on x.

Thus if L is $(t(n), q(n))$-partial-bi-immune, it is possible that there exists a machine for L which runs for less than $t(n)$ time on $q(n)$ strings of length n for every n. Every DTIME($t(n)$)-bi-immune language is $(t(n), q(n))$-partial-bi-immune for any $q(n)$. However the converse is not true. For example, for any constructible function $q(n)$ where $0 < q(n) < 2^n$, we can easily construct a language in EXP that is $(2^n, q(n))$-partial-bi-immune but not 2^n-bi-immune.

Partial Bi-immunity Hypothesis: For some $\epsilon > 0$, NP contains a $(2^{n^\epsilon}, 2^{n^{o(1)}})$-partial-bi-immune language.

In other words, the partial bi-immunity hypothesis asserts that for some $\epsilon > 0$, there is a language in NP that is $(2^{n^\epsilon}, 2^{n^\gamma})$-partial-bi-immune for all $\gamma > 0$. It is obvious that the partial bi-immunity hypothesis is weaker than the bi-immunity hypothesis. The following theorem is an improvement of the result of Pavan and Selman [23] that achieves the same conclusion under the bi-immunity hypothesis. Its proof uses substantially different techniques.

Theorem 1 (Main theorem). *If the partial bi-immunity hypothesis holds, then Turing completeness is different from many-one completeness for* NP.

Proof. Let L be a language in NP which is partial bi-immune. Let N be a nondeterministic machine for L running in time n^l for some l. Let M be the standard brute-force deterministic machine deciding L which runs in time 2^{n^l}.

In order to define the candidate language we first define segmented languages L_e, L_o, and PadSAT in the following manner. Let $k = \lfloor 10l/\epsilon \rfloor$. Let $t_1 = 2$ and $t_i = t_{i-1}^{k^2}$. Define

$$\begin{aligned} E &= \{x \mid t_i^{1/k} \leq |x| < t_i^k \text{ for even } i \ \} \\ O &= \{x \mid t_i^{1/k} \leq |x| < t_i^k \text{ for odd } i \ \} \end{aligned}$$

Note that E and O partition the set of all strings and hence for any string x, $|x| \geq 2$, x is in exactly one of E and O. Let $L_e = L \cap E$, $L_o = L \cap O$, and PadSAT $=$ SAT $\cap\, O$.

Note that L_o and L_e can be decided in deterministic time 2^{n^l} by appropriately modifying the machine M for L. Now define our Turing complete language. To keep the notation simpler, we use a three letter alphabet.

$$\mathcal{C} = 0(L_o \cup \text{PadSAT}) \cup 1(L_o \cap \text{PadSAT}) \cup 2L_o.$$

Proposition 2. *$\mathcal{C}$ is 2-Turing complete for* NP.

Proof of Proposition 2. Observe that $\mathcal{C} \in$ NP and PadSAT is many-one complete for NP. Hence it suffices to show that we can decide PadSAT by making 2 adaptive queries to $\mathcal{C}$. Given an instance x of PadSAT, if $x \in L_o$, then $x \in$ PadSAT if and only if $x \in (L_o \cap \text{PadSAT})$, else $x \in$ PadSAT if and only if $x \in (L_o \cup \text{PadSAT})$. Thus $\mathcal{C}$ is 2-Turing complete for NP. □ *Proposition 2.*

Next we will define a language in NP that is not many-one reducible to $\mathcal{C}$. We will use left sets [21] for this.

Definition. For a language $L \in \mathrm{NP}$ decided by an NP-machine N, define

$$\begin{aligned} \mathit{Left}(L) \;=\; & \{\langle x, y\rangle \mid \text{ there exists a } z \text{ such that } y \leq z \text{ and} \\ & z \text{ is an accepting computation of } N \text{ on input } x\}. \end{aligned}$$

Here $<$ is the dictionary order, with $0 < 1$. Note that $\mathit{Left}(L) \in \mathrm{NP}$.

We will show that under the partial bi-immunity assumption the language $\mathit{Left}(L_e)$ is not many-one reducible to $\mathcal{C}$. For this we will assume the contrary and show that either there exists a deterministic machine M_{L_e} that correctly decides L_e which, at infinitely many input lengths n from E, runs in time $\leq 2^{n^\epsilon}$ on *all* strings at those lengths; or there exists a machine M_{L_o} that correctly decides L_o which, at infinitely many input lengths n from O, runs in time $\leq 2^{n^\epsilon}$ for more than $2^{n^{o(1)}}$ strings. Since L_e coincides with L on strings from E, and L_o coincides with L on strings from O, this contradicts the partial bi-immunity of L.

Assume that $\mathit{Left}(L_e)$ reduces to the candidate language by a many-one reduction f running in time n^r for some r. For an input length n, let S_n denote the set $\{f(\langle x, y\rangle) \mid |x| = n, |y| \leq n^l\}$. Divide S_n into three disjoint sets as follows.

$$\begin{aligned} A_n &= \{z \in S_n \mid |z| \leq n^{\frac{1}{k}}\} \\ B_n &= \{z \in S_n \mid z \in E, z \notin A_n\} \\ C_n &= \{z \in S_n \mid z \notin A_n \cup B_n\} \end{aligned}$$

Observation 3. *C_n is a subset of O. If $n = t_{2i}$ for an i, then for every $z \in C_n$, $|z| > n^k$.*

First we state a claim that we use for the proof.

Claim 4. *One of the following holds.*

1. *$(\exists^\infty n \in \{t_{2i}\}_{i\geq 1})$ for which there exist at least $2^{n^{\epsilon/2}}$ distinct strings $\{0z_i\}_{1\leq i\leq 2^{n^{\epsilon/2}}}$ in C_n so that for each of these strings $0z_i$, $f^{-1}(0z_i) \cap \mathit{Left}(L_e) = \emptyset$*

2. *$(\exists^\infty n \in \{t_{2i}\}_{i\geq 1})$ for which there exist at least $2^{n^{\epsilon/2}}$ distinct strings $\{1z_i\}_{1\leq i\leq 2^{n^{\epsilon/2}}}$ in C_n so that for each of these strings $1z_i$, $f^{-1}(1z_i) \subseteq \mathit{Left}(L_e)$*

3. *$(\exists^\infty n \in \{t_{2i}\}_{i\geq 1})$ for which there exist at least $2^{n^{\epsilon/2}}$ distinct strings $\{2z_i\}_{1\leq i\leq 2^{n^{\epsilon/2}}}$ in C_n so that for each of these strings $2z_i$, $f^{-1}(2z_i) \cap \mathit{Left}(L_e) = \emptyset$*

Assuming Claim 4, we can finish the proof of the theorem. We now do this and defer the proof of Claim 4 until later.

We give a description of machine M_{L_o} for deciding L_o and argue that for infinitely many input lengths m from O, for more than 2^{m^γ} strings of this length, the machine runs in time $\leq 2^{m^\epsilon}$, where $\gamma = \frac{\epsilon}{4(rl+r)}$. Since L is the same as L_o on strings from O, this contradicts the partial bi-immunity of L.

MACHINE M_{L_o}(z)
$m \leftarrow |z|$; If $z \notin O$, **reject** z.
for $n = 1$ to $m^{\frac{1}{k}}$
    for all x of length n
        for all y, $|y| \leq n^l$

5 compute $f(\langle x, y\rangle)$;

6 **If** for no $\langle x, y\rangle$, $f(\langle x, y\rangle) = bz$ with $b \in \{0, 1, 2\}$

7 **then** run M on z.

8 **else** /* for some $\langle x, y\rangle$ and for some $b \in \{0, 1, 2\}$, $f(\langle x, y\rangle) = bz$ */

9 **Case** b of

10 $b = 0$: **if** $\langle x, y\rangle \notin \mathit{Left}(L_e)$

11 **reject** z

12 **else** run M on z

13 $b = 1$: **if** $\langle x, y\rangle \in \mathit{Left}(L_e)$

14 **accept** z

15 **else** run M on z

16 $b = 2$: **if** $\langle x, y\rangle \in \mathit{Left}(L_e)$

17 **accept** z, **else reject** z.

We now show that for every length n at which one of the three cases of Claim 4 holds, there exists an input length m from O such that M_{L_o} runs in less than 2^{m^ϵ} time on more than 2^{m^γ} strings of length m, where $\gamma = \epsilon/4(rl + r)$.

Observe that the above algorithm computes f only on tuples $\langle x, y\rangle$ for which $|y| \le |x|^l$. Since f is n^r time bounded, $|f(\langle x, y\rangle)| \le |x|^{r(l+1)}$. Let n be an input length for which the first part of Claim 4 holds. Since $n = t_{2i}$, by Observation 3 for every $0z \in C_n$, $|0z| > n^k$. Thus there are $2^{n^{\epsilon/2}}$ strings $0z \in C_n$ with $n^k < |0z| \le n^{r(l+1)}$ and $f^{-1}(0z) \cap \mathit{Left}(L_e) = \emptyset$. Therefore, by the pigeonhole principle, there is some input length m, $n^k < m \le n^{r(l+1)}$, where there are at least $2^{n^{\epsilon/4}} \ge 2^{m^{\epsilon/4r(l+1)}}$ strings $0z$ so that $f^{-1}(0z) \cap \mathit{Left}(L_e) = \emptyset$. Note that by definition of C_n, all strings of length m are in O.

We claim that these strings will be correctly rejected in *line* 11 of the machine. Let z be one such string. Thus there exists $\langle x, y\rangle$ such that $f(\langle x, y\rangle) = 0z$, and $\langle x, y\rangle \notin \mathit{Left}(L_e)$. Since f is a many-one reduction, we have $f(\langle x, y\rangle) \notin \mathit{Left}(L_e) \Rightarrow 0z \notin \mathcal{C} \Rightarrow z \notin L_0 \cup \text{PadSAT} \Rightarrow z \notin L_0$.

Similar arguments show that if n is a length at which the second part or the third part of the claim holds, there exists a length m from O such that M_{L_o} runs in less than 2^{m^ϵ} time on more than $2^{m^{\epsilon/4(rl+r)}}$ strings of length m.

The running time of the machine on these strings is bounded as follows. Each iteration of the *for* loop takes $2^n \times 2^{n^l+1} \times n^{r(l+1)} \times 2^{n^l}$. Since the maximum value of n is $m^{\frac{1}{k}}$ and $k = 10l/\epsilon$, the total time is bounded by $\le 2^{m^\epsilon}$.

Thus f can not be a many-one reduction from $\mathit{Left}(L_e)$ to $\mathcal{C}$. Thus $\mathcal{C}$ is not many-one complete for NP. Modulo the proof of Claim 4, we have established the theorem.

We will now prove Claim 4. For that we first need to establish an additional claim.

Claim 5. *For all strings $\langle x, y\rangle$ so that $|x| = t_{2i}$ for some i and $f(\langle x, y\rangle) \in A_n \cup B_n$ the membership of $\langle x, y\rangle$ in $\mathit{Left}(L_e)$ can be decided in deterministic time $2^{n^{\epsilon/2}}$.*

Proof of Claim 5. Let $bz = f(\langle x, y\rangle)$, where $b \in \{0, 1, 2\}$. Then if $bz \in B_n$, then it is in E, and not in $\mathcal{C}$. Hence $\langle x, y\rangle \notin \mathit{Left}(L_e)$. If $bz \in A_n$ then $|z| \le n^{\frac{1}{k}}$. We can decide the membership of bz in $\mathcal{C}$, if we know the membership of z in L_o and PadSAT. The membership of z in PadSAT can be decided in time $2^{|z|}$ and the membership in L_o can be decided in time $2^{|z|^l}$. Thus the total time taken is $2^{|z|} + 2^{|z|^l} < 2^{n^{1/k}} + 2^{n^{l/k}}$. Since $k = 10l/\epsilon$, this is less than $2^{n^{\epsilon/2}}$. Finally, since f is a many-one reduction from $\mathit{Left}(L_e)$ to $\mathcal{C}$, $\langle x, y\rangle \in \mathit{Left}(L_e) \Leftrightarrow bz \in C$. $\square$ *Claim 5.*

Proof of Claim 4. Under the assumption that the claim is not true, we will show that the machine M_{L_e} described below will correctly decide L_e. Moreover, for all but finitely many inputs lengths n of the form t_{2i}, for *all inputs* at that length, M_{L_e} will run in time $\leq 2^{n^\epsilon}$. This will contradict the partial bi-immunity of L, since L and L_e are identical on input lengths of the form t_{2i}.

For an arbitrary length n, let $0C_n$ denote the set of elements in C_n of the form $0z$. Similarly define $1C_n$ and $2C_n$. For an input x, let T_x denote the computation tree of N on input x. The machine will keep three lists l_0, l_1, l_2 where l_i will contain nodes y of T_x so that $f(\langle x, y\rangle) \in iC_n$. It will also satisfy the invariant that the size of l_i is always $\leq 2^{n^{\epsilon/2}}$. A description of the machine M_{L_e} is given below.

MACHINE $M_{L_e}(x)$
1 **if** $|x|$ is not of the form t_{2i}
2   **then** simulate the deterministic machine for L_e and decide.
3 $n \leftarrow |x|$;
4 $l_0, l_1, l_2 \leftarrow \lambda$; /* Initialize lists to the root of T_x */
5 **while** $|l_i| \leq 2^{n^{\epsilon/2}}$ for all $i = 0, 1, 2$
6    $X \leftarrow$ EXPAND$(l_0 \cup l_1 \cup l_2)$;
7   **if** $X =$ NULL /* All the nodes in the lists are leaves */
8   **then if** $N(x)$ accepts on computation path y for some $y \in l_0 \cup l_1 \cup l_2$ **accept**.
9   **else reject**. /* N is the nondeterministic machine for L */
10   **else for** each $y \in X$
11   **if** $f(\langle x, y\rangle) \in A_n \cup B_n$
12   simulate $2^{n^{\epsilon/2}}$-machine for deciding $\langle x, y\rangle \in \mathit{Left}(L_e)$; /* Claim 5 */
13   **if** $\langle x, y\rangle \in \mathit{Left}(L_e)$ **accept.** /* $\exists z \leq y$ which is an accepting path of N */
14   **else** $X \leftarrow X \setminus \{y\}$;
15   **if** $X = \phi$ **reject**.
16          $X \leftarrow$SHORTEN(X);
17   **for** i=0,1,2 place y from X in list l_i if and only if $f(\langle x, y\rangle) \in iC_n$
18 **end-while**
19 If $|l_0| > 2^{n^{\epsilon/2}}$ or $|l_2| > 2^{n^{\epsilon/2}}$ then **accept**.
20 If $|l_1| > 2^{n^{\epsilon/2}}$ then PRUNE(l_1);
21 **goto** line 5; /* while loop */

EXPAND takes a set of nodes in the computation tree and outputs all their children. If all the nodes in the set are leaves, then EXPAND returns NULL. SHORTEN takes a set X of nodes, and for two strings $y_1, y_2 \in X$, if $y_1 < y_2$ and $f(\langle x, y_1\rangle) = f(\langle x, y_2\rangle)$ then it discards y_1 from X. Thus SHORTEN removes redundancies in the set. PRUNE takes a set of nodes and returns the first (in the dictionary order) $2^{n^{\epsilon/2}}$ nodes in the set and discards all others.

We will first argue that M_{L_e} decides L_e correctly. Since for inputs at lengths other than $\{t_{2i}\}$, M_{L_e}'s behavior is identical to that of the deterministic machine for L_e, it decides correctly on all these inputs. For deciding inputs at length t_{2i}, it uses the structure of left-cuts: $\langle x, y\rangle \in \mathit{Left}(L_e) \Rightarrow \forall z \leq y\ \langle x, z\rangle \in \mathit{Left}(L_e)$.

Consider an input x of length $n = t_{2i}$ for some arbitrary i. We first claim that if M_{L_e} accepts x, then $x \in L_e$. If M_{L_e} accepts in *line* 8, then it found an accepting computation of N on x. So x indeed belongs to L_e. From now on assume that it accepts in *line* 19.

Since we assumed that Claim 4 is false, we have that $\forall^\infty(n \in t_{2i})$,

$$|f(\overline{Left(L_e)}) \cap 0C_n| \leq 2^{n^{\epsilon/2}}, \tag{1}$$
$$|f(\overline{Left(L_e)}) \cap 2C_n| \leq 2^{n^{\epsilon/2}}. \tag{2}$$

Consider *line* 19 of the machine. Suppose $|l_0| > 2^{n^{\epsilon/2}}$. This means that $|f(\{\langle x,y\rangle | y \in l_0\}) \cap 0C_n| > 2^{n^{\epsilon/2}}$. Hence from the inequality 1 and the fact that f is a reduction, it follows that there exists $y \in l_o$ such that $\langle x,y\rangle \in Left(L_e)$. Therefore $x \in L_e$. Similarly we can argue for the case $|l_2| > 2^{n^{\epsilon/2}}$. Hence the decision made by M_{L_e} at *line* 19 is correct.

Next we argue that if $x \in L_e$, then M_{L_e} accepts x. For this, it suffices to show that if M_{L_e} does not accept x in *line* 19, then it accepts in *line 8*. We claim that after every iteration of the *while* loop, the rightmost accepting computation of N on x passes through a node in $l_0 \cup l_1 \cup l_2$. Initially, $l_0 \cup l_1 \cup l_2$ contains the root of T_x, thus the claim is true initially. Assume that the claim is true after the $(k-1)^{\text{th}}$ iteration of the loop. Consider the k^{th} iteration of the *while* loop. Line 6 places all the children of nodes in $l_0 \cup l_1 \cup l_2$ in X, thus the rightmost accepting computation passes through a node in X. We delete a node y from X in *line* 14, only if $y \notin Left(L_e)$. Thus the rightmost accepting computation can not pass through y. Suppose the procedure SHORTEN removes a node y_1 from X. This happens only if there exists y_2 such that $y_1 < y_2$ and $f(\langle x, y_1\rangle) = f(\langle x, y_2\rangle)$. In this case either both $\langle x, y_1\rangle$ and $\langle x, y_2\rangle$ belong to $Left(L_e)$ or both of them do not belong to $Left(L_e)$. Thus the rightmost accepting computation can not pass through y_1. If after *line* 17, $|l_i| \leq 2^{n^{\epsilon/2}}$ for every i, then the k^{th} iteration of the *while* loop ends here, so X contains a node through which the rightmost accepting computation passes through. On the other hand, if $|l_1| > 2^{n^{\epsilon/2}}$, then the algorithm performs PRUNE(l_1). Note that since we assumed the algorithm does not halt in *line 19*, this is the only possibility.

Assume that PRUNE(l_1) deletes some nodes from X. Since we assumed Claim 4 is false, it holds that

$$|f(Left(L_e)) \cap 1C_n| \leq 2^{n^{\epsilon/2}}. \tag{3}$$

Since $|l_1| > 2^{n^{\epsilon/2}}$, this means that $|f(\{\langle x,y\rangle | y \in l_1\}) \cap 1C_n| > 2^{n^{\epsilon/2}}$. From the inequality (3) and the fact that f is a reduction, there are $|l_i| - 2^{n^{\epsilon/2}}$ strings y such that $\langle x,y\rangle \notin Left(L_e)$. From the structure of left-cuts it follows that all but the first $2^{n^{\epsilon/2}}$ are not in $Left(L_e)$. Hence we can discard these elements from l_1 since none of their children will lead to an accepting computation of N. Thus after the k^{th} iteration of the *while* loop the rightmost accepting computation passes through a node in X.

Since the depth of T_x is n^l, the **while**-loop is executed at most n^l times. For each iteration of the loop, $|X| \leq 6 \times 2^{n^{\epsilon/2}}$. The maximum running time needed for each of the elements in X is when *line* 12 is executed which is $O(2^{n^{\epsilon/2}})$. Therefore the running time is bounded by $O(n^l \times 2^{n^{\epsilon/2}} \times 2^{n^{\epsilon/2}}) \leq 2^{n^\epsilon}$.

Thus M_{L_e} takes less than 2^{n^ϵ} time on every string from $\cup_{n=t_{2i}} \Sigma^n$. This contradicts the partial bi-immunity of L. Thus Claim 4 is true. □ *Claim 4.*

□ *Theorem 1.*

4 Scaled Dimension

In this section, we use Theorem 1 and previous work to derive the separation of NP-completeness notions from a scaled dimension hypothesis. We refer to [9, 8] for definitions and background of scaled dimension. For this paper, all we need is the following immediate consequence of Theorem 5.2 in [8].

Theorem 6. *For all $c \in \mathbb{N}$ and $\alpha \in [0,1]$, the class of all languages that are not $(2^{cn}, 2^{2^{(\log n)^{1-\alpha}}})$-partial-bi-immune has -3rd-order scaled p-dimension at most α.*

Corollary 7. *For all $c \in \mathbb{N}$, the class of all languages that are not $(2^{cn}, 2^{n^{o(1)}})$-partial-bi-immune has -3rd-order scaled p-dimension 0.*

Corollary 8. *If the -3rd-order scaled p-dimension of* NP *is positive, then the partial bi-immunity hypothesis holds.*

The following theorem is immediate from Theorem 1 and Corollary 8.

Theorem 9. *If the -3rd-order scaled p-dimension of* NP *is positive, then Turing completeness is different from many-one completeness for* NP*.*

5 Conclusions

In this section we mention the hypotheses studied in the context of separating completeness notions and compare them. Consider the following hypotheses.

- (Measure) NP does not have p-measure 0.
- (Generic) NP contains a p-generic language.
- (Bi-immune) For some $\epsilon > 0$, NP contains a 2^{n^ϵ}-bi-immune language.
- (Partial) For some $\epsilon > 0$, NP contains a $(2^{n^\epsilon}, 2^{n^{o(1)}})$-partial-bi-immune language.
- (Scaled) The -3rd-order scaled p-dimension of NP is positive.
- (Separation) Turing completeness is different from many-one completeness for NP.

We currently know that

$$\text{(Measure)} \Rightarrow \text{(Generic)} \Rightarrow \text{(Bi-immune)} \Rightarrow \text{(Partial)} \Rightarrow \text{(Separation)}$$

and

$$\text{(Measure)} \Rightarrow \text{(Scaled)} \Rightarrow \text{(Partial)} \Rightarrow \text{(Separation)}.$$

For future work, it would be interesting to see if we can replace the partial bi-immune language in our result by a language that is hard on average. Another open problem is whether the separation of completeness notions can be obtained from a -2nd-order scaled dimension hypothesis on NP.

References

[1] K. Ambos-Spies and L. Bentzien. Separating NP-completeness under strong hypotheses. *Journal of Computer and System Sciences*, 61(3):335–361, 2000.

[2] J. Balcázar and U. Schöning. Bi-immune sets for complexity classes. *Mathematical Systems Theory*, 18(1):1–18, June 1985.

[3] H. Buhrman, S. Homer, and L. Torenvliet. Completeness notions for nondeterministic complexity classes. *Mathematical Systems Theory*, 24:179–200, 1991.

[4] H. Buhrman and L. Torenvliet. On the structure of complete sets. In *9th IEEE Annual Conference on Structure in Complexity Theory*, pages 118–133, 1994.

[5] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[6] J. Grollmann and A. L. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–355, April 1988.

[7] J. M. Hitchcock. MAX3SAT is exponentially hard to approximate if NP has positive dimension. *Theoretical Computer Science*, 289(1):861–869, 2002.

[8] J. M. Hitchcock. Small spans in scaled dimension. *SIAM Journal on Computing*, 34(1):170–194, 2004.

[9] J. M. Hitchcock, J. H. Lutz, and E. Mayordomo. Scaled dimension and nonuniform complexity. *Journal of Computer and System Sciences*, 69(2):97–122, 2004.

[10] J. M. Hitchcock and A. Pavan. Hardness hypotheses, derandomization, and circuit complexity. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 336–347. Springer-Verlag, 2004.

[11] S. Homer. Structural properties of nondeterministic complete sets. In *Proceedings of the 5th Annual IEEE Annual Conference on Structure in Complexity Theory*, pages 3–10, 1990.

[12] S. Homer. Structural properties of complete problems for exponential time. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 135–153. Springer-Verlag, 1997.

[13] S. Homer, S. Kurtz, and J. Royer. On 1-truth-table-hard languages. *Theoretical Computer Science*, 115(2):383–389, 1993.

[14] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[15] K. Ko and D. Moore. Completeness, approximation and density. *SIAM Journal on Computing*, 10(4):787–796, Nov. 1981.

[16] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[17] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9:265–266, 1973. English translation of original in *Problemy Peredaci Informatsii*.

[18] L. Longpré and P. Young. Cook reducibility is faster than Karp reducibility. *Journal of Computer and System Sciences*, 41:389–401, 1990.

[19] J. H. Lutz. Dimension in complexity classes. *SIAM Journal on Computing*, 32(5):1236–1259, 2003.

[20] J. H. Lutz and E. Mayordomo. Cook versus Karp-Levin: Separating completeness notions if NP is not small. *Theoretical Computer Science*, 164:141–163, 1996.

[21] M. Ogiwara and O. Watanabe. On polynomial time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal on Computing*, 20(3):471–483, 1991.

[22] A. Pavan. Comparison of reductions and completeness notions. In L. Hemaspaandra, editor, *SIGACT News*, Complexity Theory Column 40. ACM Press, June 2003.

[23] A. Pavan and A. L. Selman. Bi-immunity separates strong NP-completeness notions. *Information and Computation*, 188(1):116–126, 2004.

[24] A. L. Selman. Reductions on NP and P-selective sets. *Theoretical Computer Science*, 19:287–304, 1982.

[25] O. Watanabe. A comparison of polynomial time completeness notions. *Theoretical Computer Science*, 54:249–265, 1987.