On Closeness of NP-hard Sets to other Complexity Classes¹

Bin Fu
Beijing Computer Institute
Beijing, 100044, P.R.China

Hong-zhou Li²
Department of Mathematics
Yunnan Education College
Kuming, 600031, P.R.China

Abstract

Let A be a language and C be a class of languages. A is said s - C-close (s - C-outside-close) if there exists B ∈ C such that \( (A \triangle B)^{s(n)} \leq s(n) \) (and \( A \subseteq B \)). If A is q - C-close (q - C-outside-close) for some polynomial q then we simply call A is C - close (C - outside - close). The following results are shown in this paper.

1) No NP-hard set can be co - NP-close unless \( NP = co - NP \).

2) No NP-hard set can be R-close unless \( NP = R \).

3) No NP-hard set can be O(log n) - UP-close unless \( NP \subseteq FewP \).

4) No NP-hard set can be O(log n) - C=P-close unless \( NP \subseteq C=P \).

5) No NP-hard set can be UP - outside - close unless \( NP = FewP \).

6) No NP-hard set can be C=P - outside - close unless \( NP \subseteq C=P \).

1.Introduction

Research about the connection and difference between complexity classes are two main research lines in complexity theory. Recent years investigations about many counting classes revealed the surprising connections between complexity classes. In this paper we pay our attention to the difference between NP and other complexity classes. We study whether a NP-hard set can be approximated sufficiently by the sets in other complexity classes.

Yesha [17] first considered to measure the similarity of two sets A and B by the density of their symmetric difference \( A \triangle B \). Two sets A and B are said s - close to each other was defined by Yesha [17] and Schöning [10] such that if for each n, \( (A \triangle B)^{s(n)} \leq s(n) \).

Can a NP-hard set be polynomially close to any set in \( P \) unless \( P = NP \). Schöning showed that no paddable NP-hard sets can be polynomially close to any set in \( P \) unless \( P = NP \). Watanabe [16] showed that if \( NP \subseteq P_{1\cdot44}(Sparse) \) then \( NP = R \). The question was not settled until Ogiwara and Watanabe [1] showed that if \( NP \subseteq P_{1\cdot44}(Sparse) \) then \( P = NP \).

Recently Fu [8] investigated lower bounds of closeness between many complexity classes. He showed that if a NP-hard set is the union of a set in \( P_{1\cdot44}(Sparse) \) and set A, then \( NP \subseteq P_{1\cdot44}(A) \). Thus no NP-hard set can be the union of a sparse set and a set in \( co - NP \) unless \( NP = co - NP \) (\( NP = FewP \)).

In this paper, we investigate the closeness between NP-hard sets and the sets of some other complexity classes such as \( UP, co - NP, C=P, \) and \( R \). In order to characterize the necessary conditions for a NP-hard set which has small density symmetric difference with a set A, we introduce two reductions: \( \leq_{d-maj}^P \) and \( \leq_{d-conj}^P \), both of which are positive truth-table reductions [11]. We show that if a NP-hard set can be polynomially close \( O(log n) - close \) to set A, then \( NP \subseteq P_{d-maj}(A) \) (\( NP \subseteq P_{d-conj}(A) \) resp.), and if \( NP - hard \) set H is the subset of A and A - H is sparse, then \( NP \subseteq P_{d-conj}(A) \).

1This research is supported in part by HTP863. [6] is the earlier version of this paper.
2This research was performed while this author was visiting Beijing Computer Institute.

0-8186-2925-X/92 $03.00 © 1992 IEEE
2. Preliminaries

We fix \( \Sigma = \{0, 1\} \) as our alphabets. By "string" we mean an element of \( \Sigma^* \). For a string \( x \in \Sigma^* \), \( |x| \) denotes the length of \( x \). We consider a standard canonical order on \( \Sigma^* \). For any strings \( x \) and \( y \), \( x \leq y \) denotes the relation with respect to lexicographic order.

Now we involve the following complexity classes:

- **\( \text{P} \)**: Languages accepted by deterministic polynomial-time Turing machines.
- **\( \text{NP} \)**: Languages accepted by nondeterministic polynomial-time Turing machines.
- **\( \text{UP} \)**: Languages accepted by polynomial-time nondeterministic Turing machines.
- **\( \text{co-NP} \)**: Languages accepted by polynomial-time nondeterministic Turing machines.

Our computation model is the Turing machine. We use the pairing function \( \langle \cdot, \cdot \rangle \). We often need to consider polynomial-time computable functions. We use the pairing function \( \langle \cdot, \cdot \rangle \) to denote the null string. For example, \( \langle 1, 2 \rangle = \langle 3, 4 \rangle \).

We now define some notions of polynomial-time reducibilities.

A \( \leq_{\text{p}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( x \in A \iff f(x) \in B \).

A \( \leq_{\text{tt}} \) reduction of \( A \) to \( B \) is a pair \( < f, g > \) of polynomial-time computable functions such that the following hold for each \( x \in \Sigma^* \):

1. \( f(x) = \langle x_1, ..., x_k \rangle \) is an ordered \( k \)-tuple of strings,
2. \( g(x) \) is a \( k \)-argument truth-table: \( \{0, 1\}^k \rightarrow \{0, 1\} \),
3. \( x \in A \iff g(x)(\chi_B(x_1), ..., \chi_B(x_k)) = 1 \), where \( \chi_B \) is the characteristic function of the set \( B \).

A \( \leq_{\text{tt}} \) reduction of \( A \) to \( B \) is a \( \leq_{\text{p}} \) reduction of \( A \) to \( B \) for some integer \( k \).

A \( \leq_{\text{adj}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( f(x) = \langle x_1, ..., x_k \rangle \) and \( x \in A \iff x_i \in B \) for some \( 1 \leq k \).

A \( \leq_{\text{adj}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( f(x) = \langle x_1, ..., x_k \rangle \), and \( x \in A \iff x_i \in B \) for some \( 1 \leq k \).

A \( \leq_{\text{adj}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( f(x) = \langle x_1, ..., x_k \rangle \), and \( x \in A \iff x_i \in B \) for some \( 1 \leq k \).

A \( \leq_{\text{adj}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( f(x) = \langle x_1, ..., x_k \rangle \), and \( x \in A \iff x_i \in B \) for some \( 1 \leq k \).

A \( \leq_{\text{adj}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( f(x) = \langle x_1, ..., x_k \rangle \), and \( x \in A \iff x_i \in B \) for some \( 1 \leq k \).

A \( \leq_{\text{adj}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( f(x) = \langle x_1, ..., x_k \rangle \), and \( x \in A \iff x_i \in B \) for some \( 1 \leq k \).

A \( \leq_{\text{adj}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( f(x) = \langle x_1, ..., x_k \rangle \), and \( x \in A \iff x_i \in B \) for some \( 1 \leq k \).

A \( \leq_{\text{adj}} \) reduction of \( A \) to \( B \) is a polynomial-time computable function \( f \) such that for each \( x \in \Sigma^* \), \( f(x) = \langle x_1, ..., x_k \rangle \), and \( x \in A \iff x_i \in B \) for some \( 1 \leq k \).
n ∈ N. Let language A, B ⊆ Σ*, we define A △ B = (A − B) ∪ (B − A). The function dist_{A,B} : N → N is called the distance function of A and B, where dist_{A,B}(n) = || (A △ B)^S_n ||. Let C be a class of languages and function s : N → N, A is said s-C-close (s - C - outside - close, s - C - inside - close) if there exists B ∈ C such that dist_{A,B}(n) ≤ s(n) (and A ⊆ B, B ⊆ C resp.). If A is q - C - close (q - C - outside - close, q - C - inside - close) for some polynomial q then we simply call A is C - close (C - outside - close, C - inside - close). We assume all polynomials involved in this paper are monotonic.

### 3. Closeness to NP - m - hard sets.

In this section we study whether a NP - hard set can be s(n) - close to some other complexity classes for some slowly increasing function s(n). First we develop a useful technical lemma which generalizes many results.

**Lemma 3.1.** Let H be NP - m - hard, K ⊆ NP, A ⊆ Σ* and dist_{A,H}(n) ≤ s(n), where s(n) is a time constructible monotonic function. There exists a polynomial p_0(n) such that for every time constructible function h(n) > 2(p_0(n)) there is an algorithm which has the following properties for input x of length n:

1) The algorithm either accepts x or outputs a series of sets: G_1, ..., G_{s(n)}, where ∥ G_e ∥ ≤ h(n) and G_e ⊆ Σ^{≤p_0(n)} for all e ≤ e_0.

2) x ∈ K ⇐⇒ (The algorithm accepts x) or (∥ A ∩ G_e ∥ > s(p_0(n)) for some e ≤ e_0).

3) If ∥ A ∩ G_e ∥ > s(p_0(n)) for all e with ∥ G_e ∥ = h(n), then x ∈ K ⇐⇒ the algorithm accepts x.

4) The algorithm will stop in p_1(h(n)) + p_1(n) steps, where p_1(n) is a polynomial.

**Proof.** Since K ⊆ NP, there exist a set C ∈ P and a polynomial r(n) such that x ∈ K ⇐⇒ ∃w(w ∈ Σ^{≤r(|x|)} ∧ < x, w > ∈ C). We define the left set L(C, r) like that in [9]:

L(C, r) = {< x, y > : (y ∈ Σ^{≤r(|x|)}) ∧ (∃w ∈ Σ^{≤r(|x|)}(y ≤ w)) ∧ (< x, w > ∈ C)}.

It is easy to see L(C, r) ∈ NP. Because H is NP - m - hard, there exists a function f ∈ PF such that L(C, r) ≤_m H via f. Let f be computable in polynomial time t(n). For a fixed x ∈ Σ*, if there exists a w ∈ Σ^{≤r(|x|)} such that < x, w > ∈ C, then we let w_{max} be the largest one.

Let p_0(n) = t(2(n + r(n))).

**Algorithm:**

Input x of length n.

Let U_0 := {(λ, 1^{r(n)})}. e := 0.

Repeat

Let [a_1, b_1], [a_2, b_2], ..., [a_e, b_e] be all the intervals in U_e.

U_e is divided into the following divisions : U_{e,1}, U_{e,2}, ... which satisfy three conditions as follows:

1) U_e = U_{e,1} ∪ U_{e,2} ∪ ... ∪ U_{e,m}, where m_e is the number of divisions.

2) Any two intervals [a, b], [a', b'] are in the same division if and only if f(< x, a >) = f(< x, a' >).

3) For any two divisions U_{e,i}, U_{e,j}, i < j, [a^{(i)}, b^{(i)}] < [a^{(j)}, b^{(j)}]. Where [a^{(i)}, b^{(i)}] and [a^{(j)}, b^{(j)}] are the least intervals (leftmost intervals) of U_{e,i} and U_{e,j} respectively.

Let [a^{(1)}, b^{(1)}] be the largest interval (rightmost interval) in U_{e,i} (i ≤ m_e).

V_e := {[a^{(i)}, b^{(i)}] : i ≤ h(n)}

G_e := {f(< x, a^{(i)} >) : a^{(i)} is the left point of one of the intervals in V_e}

U_{e+1} := \bigcup_{v \in G_e} \{t_1, t_2 : t_1, t_2 are the first and last half of t respectively\}

e := e + 1

Until all the intervals in U_e are of width 1.

e_e := e - 1 {For e was added an extra one before exiting the cycle}

If there is b ∈ \bigcup_{i \in U_e} s such that < x, b > ∈ C.

Then Accept x.

Else Output: G_1, ..., G_{s(n)}.

End of the Algorithm.

**Claim 1 i.** For all e ≤ e_s, there are at most
2h(n) intervals in $U_e$. ii). The cycle of the algorithm will not be repeated more than r(n) + 1 times. iii). $G_e \subseteq \Sigma^{=\leq h(n)}$ and $\|G_e\| \leq h(n)$. iv). $\|G_e\| = \|V_e\|$.

**Proof of Claim 1** From the algorithm i),ii) and iv) are hold clearly. We only proof iii). For each interval $[c_i, d_i]$ in $V_e$, $[c_i, d_i] \subseteq [\lambda, \gamma(n)]$. Hence $|c_i| \leq r(n)$ and $|<x, c_i>| \leq 2(n+1 - c_i) \leq 2(n + r(n))$. Because f is computable in t time, $f(<x, c_i>) \leq t(2(n + r(n))) = p_0(n)$. So $G_e \subseteq \Sigma^{=\leq h(n)}$.

**Claim 2** If for some $e \leq e_0$, $\|G_e \cap A\| > s(p_0(n))$ then $x \in K$.

**Proof Claim 2.** Clearly for any two different intervals $[c_i, d_i], [c_j, d_j]$ in $V_e$, they belong to different divisions $U_{c_i}, U_{c_j}$ respectively. Thus $f(<x, c_i>) \neq f(<x, c_j>)$. By Claim 1 iii) $G_e \subseteq \Sigma^{=\leq h(n)}$. Since $\|(A \Delta H) \leq \Sigma^{=\leq h(n)}\| \leq s(p_0(n))$, and $\|G_e \cap A\| > s(p_0(n))$. Therefore if $G_e \neq \emptyset$, and there are some $c_i \in \Sigma^{=\leq h(n)}$ with $f(<x, c_i>) \in H$. So $x \in K$.

**Claim 3** Let $e \leq e_0$. If $\|G_e \| < h(n)$, then $w_{max}$ is in an interval of $U_e$ if $w_{max}$ is in an interval of $V_e$.

**Proof of Claim 3.** We assume that $w_{max}$ is in one of the intervals of the division $U_{c_{i_0}}$ ($i_0 \leq m_e$). For any two intervals $[c, d], [c', d']$ in the same division, $f(<x, c>) = f(<x, c'>)$. So $w_{max}$ must be in $[c_{i_0}, d_{i_0}]$ which is the largest interval in $U_{c_{i_0}}$.

By Claim 1 iv) $\|V_e\| \leq \|G_e\| < h(n)$, thus $\|V_e\| \leq m_e$. For every division $U_{c_{i_0}}$ ($i \leq m_e$), its largest interval $[c_{i_0}, d_{i_0}]$ belongs to $V_e$. Hence $w_{max}$ belongs to an interval in $V_e$.

**Claim 4** Let $e \leq e_0$. If $\|G_e \cap A\| < s(p_0(n))$, then $w_{max}$ is in an interval of $U_e$ if $w_{max}$ is in an interval of $V_e$.

**Proof of Claim 4.** If $\|G_e\| < h(n)$, this claim holds by Claim 3. We only discuss the case $\|G_e\| = h(n)$. By Claim 1 iv), $\|G_e\| = \|V_e\|$. We assume that $w_{max}$ is in one of the intervals of the division $U_{c_{i_0}}$ ($i_0 \leq m_e$). By the same reasons as Claim 3 $w_{max}$ must be in $[c_{i_0}, d_{i_0}]$. By Claim 1 iii), $G_e \subseteq \Sigma^{=\leq h(n)}$. Since $\|(A \Delta H) \leq \Sigma^{=\leq h(n)}\| \leq s(p_0(n))$ and $\|G_e \cap A\| > s(p_0(n))$. Hence $H \setminus G_e \neq \emptyset$, and there are some intervals $[c_{i_1}, d_{i_1}]$ in $V_e$ such that $f(<x, c_{i_1}>) \not\in H$. Thus we have $i_1 \leq h(n)$ with $f(<x, d_{i_1}>) \not\in H$ and $w_{max} < a_{i_0}$. So $i_0 < i_1 \leq h(n)$ and $w_{max}$ belongs to an interval in $V_e$.

**Claim 5** If $\|G_e \cap A\| > s(p_0(n))$ for all $e \leq e_0$ with $\|G_e\| = h(n)$, then $x \in K$ if and only if the algorithm accept $x$.

**Proof of Claim 5.** By the algorithm and Claim 4.

**Claim 6** $x \in K$ if and only if the algorithm accepts $x$ or $\|G_e\| = h(n)$ for some $e \leq e_0$.

**Proof of Claim 6.** By algorithm and Claim 2 it's easy to see.

Let $x \in K$. We assume that the algorithm does not accept $x$.

It's impossible that $\|G_e\| < h(n)$ for all $e \leq e_0$, otherwise the algorithm will accept $x$ by Claim 3.

It's also impossible that $\|A \cap G_e\| > s(p_0(n))$ for all $e \leq e_0$ with $\|G_e\| = h(n)$, otherwise the algorithm will accept $x$ by Claim 5.

Therefore there exists an $e \leq e_0$ such that $\|A \cap G_e\| < s(p_0(n))$ and $\|G_e\| = h(n)$. Hence $\|A \cap G_e\| = \|G_e\| > \|A \cap G_e\| + s(p_0(n)) > s(p_0(n))$.

The following claim is easy to verify from the algorithm.

**Claim 7** For some polynomial $p_1(n)$, the algorithm will stop in $p_1(h(n)) + p_1(n)$ steps.

This finishes the proof of Lemma 3.1.

Now we state several consequences of our main lemma, which examine the difference $NP$ and other complexity classes.

**Theorem 3.2** Let $H$ be $NP - m$-hard. If $H \Delta A$ is sparse, then $NP \subseteq P_{d=m+1}(A)$.

**Proof.** Let $K$ be $NP - m$-complete and $s(n)$ be a polynomial such that $dist_{A,H}(n) \leq s(n)$. By Lemma 3.1 we get a polynomial $p_0(n)$. Let $h(n) =$
It is well known that the class co - NP is closed under positive truth-table reductions \([11]\). Since \(\leq_d\text{-maj}\) is a special kind of positive reduction. So we have the following result.

**Corollary 3.3** Let \(H\) be \(NP - m - hard\). If \(H\) is \(co - NP\) - close, then \(NP = co - NP\) \(\Box\)

Unfortunately we don't know whether \(UP, FewP\), or \(C_mP\) is closed under \(\leq_d\text{-maj}\) - reductions. Thus, we can't obtain the similar results for \(UP, FewP\), or \(C_mP\).

It is easy to see that \(R\) is closed under positive truth-table reductions. So we also have following corollary.

**Corollary 3.4** Let \(H\) be \(NP - m - hard\). If \(H\) is \(R\) - close, then \(NP = R\) \(\Box\)

Unfortunately we don't know whether \(UP, FewP\), or \(C_mP\) is closed under \(\leq_d\text{-maj}\) - reductions. Thus, we can't obtain the similar results for \(UP, FewP\), or \(C_mP\).

**Theorem 3.5** Let \(H\) be \(NP - m - hard\) and \(A \subseteq \Sigma^*\). If \(dist_{A,H}(n) = O(log(n))\), then \(NP \subseteq \Pi_{d\text{-conj}}(A)\).

**Proof.** Let \(K\) be \(NP - m - complete\). Since \(||H \cap A|| \leq c \cdot log(n) + c\) for some \(c > 0\). Let \(s(p_0(n)) = c \cdot log(n) + c\), we use Lemma 3.1 and let \(h(n) = 2s(p_0(n)) + 1\). Clearly \(h(n) = O(log(n))\). By Lemma 3.1.2 we have: \(x \in K \iff (The\ algorithm\ accepts\ x)\) or \(||G_s \cap A|| > s(p_0(n))\) for some \(c \leq e_\alpha\). Since \(||G_s \cap A|| > s(p_0(n))\) \(\iff \exists G((G \subseteq G_s) \wedge (||G|| = s(p_0(n)) + 1) \wedge (G \subseteq A))\).

So, \(||G_s \cap A|| > s(p_0(n))\) for some \(c \leq e_\alpha\) \(\iff \exists e_\alpha \exists G((G \subseteq G_s) \wedge (||G|| = s(p_0(n)) + 1) \wedge (G \subseteq A))\).

Since \(||G_s|| = O(log(n))\), the number of subsets of \(G_s\) is bounded by polynomial. Therefore \(K \in \Pi_{d\text{-conj}}(A)\).

Hence \(NP \subseteq \Pi_{d\text{-conj}}(A)\) \(\Box\)

**Corollary 3.6** Let \(H\) be \(NP - m - hard\). If \(H\) is \(O(log(n)) - UP\) - close, then \(NP = FewP\) \(\Box\)

**Lemma 3.7** \(C_mP\) is closed under \(\leq_d\text{-maj}\) - reductions.

**Proof.** We use the techniques of low-degree polynomial of \([3]\) and \([4]\). Note that \(\forall((\bar{z}_1,0) \wedge ... \wedge (\bar{z}_k,0)) \iff \prod(x_i^2 ... x_k^2) = 0\) \(\Box\)

**Corollary 3.8** Let \(H\) be \(NP - m - hard\). If \(H\) is \(O(log(n)) - C_mP\) - close, then \(NP \subseteq \Pi_{d\text{-conj}}(A)\).

**Theorem 3.9** Let \(H\) be \(NP - m\)-hard and \(A\) be a language. If \(H \subseteq A\) and \(A - H\) is sparse then \(NP \subseteq \Pi_{d\text{-conj}}(A)\).

**Sketch of Proof.** The proof of this theorem is very similar to that of Lemma 3.1 except let \(h(n) = s(p_0(n)) + 1\) and replace the Claim 4 and Claim 6 by the following Claim 4' and Claim 6' respectively.

**Claim 4'** Let \(e \leq e_\alpha\). If \(G_s \cap A \neq \emptyset\), then \(w_{max}\) is in an interval of \(U_e \iff w_{max}\) is in an interval of \(V_e\).

**Proof of Claim 4'.** Since \(G_s \cap A \neq \emptyset\) and \(H \subseteq A\), therefore we have some interval \([c_i, d_i]\) in \(V_e\) such that \(f(<z, a_i>) \notin H\). Like in Claim 4 \(w_{max}\) is in an interval of \(U_e \iff w_{max}\) belongs to an interval in \(V_e\).

**Claim 6'** \(x \in K \iff (The\ algorithm\ accepts\ x)\) or \((G_s \subseteq A)\) and \(G_s \models h(n)\) for some \(e \leq e_\alpha\).

From Theorem 3.9 , Lemma 3.6 and Lemma 3.9 we have:

**Corollary 3.10** Let \(H\) be \(NP - m - hard\). If \(H\) is \(UP - outside\) - close, then \(NP = FewP\) \(\Box\)

**Corollary 3.11** Let \(H\) be \(NP - m - hard\). If \(H\) is \(C_mP - outside\) - close, then \(NP \subseteq \Pi_{d\text{-conj}}(A)\).

We left some open problems:
1. Does \( NP \neq co-NP(NP \neq R) \) implies that for any \( NP - m\) hard set \( H \) and \( A \in co-NP(R) \) resp., \( A \triangle H \) is not in \( P_{\text{set}}(\text{Sparse}) \)?

2. Does \( \text{FewP} \neq NP \) implies that for any \( NP - m\) hard set \( H \) and \( A \in \text{FewP}, A \triangle H \) is not Sparse?

3. Does \( UP \neq NP \) implies that for any \( NP - m\) hard set \( H \), \( H \) is not \( UP - \text{inside} - \text{close} \)?

Acknowledgement

We would like to thank Prof. Shouwen Tang for his encouragement in this research. The first author would also like to thank Prof. Qiongshang Li for his encouragement during this research and Mr. Tian Liu for his valuable suggestions for the earlier version of this paper [6]. We thank Mr. Anjiang Ma for his patience in typing this paper.

References


