The Self-Reducibility Technique

Qi Ge   Xiaoming Gu

Department of Computer Science
University of Rochester

November 9, 2006
Outline

1 Motivation
2 Definitions
3 Tally NP-complete Languages
4 Sparse coNP-complete Language
5 Sparse NP-complete Language
6 Sparse languages in NP-P
   - E and NE
   - Theorem
   - Summary
Motivation

- Is there a sparse coNP-complete language?
- Is there a sparse NP-complete language?
- Is there a sparse language in NP−P?
Self-Reducibility

**Definition**

A language $L$ is said to be self-reducible, if there is a deterministic polynomial-time TM $M$ such that $L = L(M^L)$ and, for each input of length $n$, $M^L(x)$ queries the oracle for words of length, at most, $n - 1$.

**Example**

**SAT**:

- $SAT = \{ F | F \text{ is satisfiable.} \}$

- A formula $F$ with variables $v_1, v_2, \ldots, v_n$ is satisfiable iff either $F[v_i = \text{True}]$ or $F[v_i = \text{False}]$ is satisfiable, for each $1 \leq i \leq n$. 
Motivation

Definitions

Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
The Self-Reducibility Technique

Qi Ge and Xiaoming Gu
Motivation

Definitions

Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
Sparse Sets

**Definition**
A set $S$ is sparse if there is some polynomial $p$ such that $|S^{\leq n}| = |\{x | x \in S \land |x| \leq n\}| \leq p(n)$.

**Example**
Tally set:
- $T \subseteq \{1\}^*$
- $p(n) = n + 1$
- There is some tally set which is even undecidable, e.g., $T_{HP} = \{1^{(1x)}_{binary} | x \in HP\}$. 
Tally NP-complete Languages

- An easier problem: is there any tally NP-complete language?
- Tally set: $T \subseteq \{1\}^*$. 

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
Theorem

If there is a tally set that is \( \leq_p^m \)-hard for NP, then P=NP.
Tally NP-complete Languages (Cont.)

How to prove?

- $\text{SAT} \leq^p_m T$, that is there is a polynomial-time deterministic function $g$ many-one reducing $\text{SAT}$ to $T$.
- $T \subseteq \{1\}^*$. 
- $\text{SAT}$ is 2-disjunctive-self-reducible.
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

The Self-Reducibility Technique

F

F is not satisfiable

F[v1=True]
F[v1=False]

Σ*
T = \{1\}^*
g(F)

Qi Ge and Xiaoming Gu
Motivation

Definitions

Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

\[ F \]

\[ F[v_1 = \text{True}] \]

\[ F[v_1 = \text{False}] \]

Not satisfiable

Collide, no need

\[ T = \{1\}^* \]

\[ T \leq p_x(|F|) \]

Keep it

\[ \Sigma^* \]
Tally NP-complete Languages (Cont.)

Algorithm

Input: $F$ with $v_1, v_2, \ldots, v_n$ as its variables

1. Initialization: $C = \{F\}$;
2. Do $i = 1$ to $n$
3. Expand $C$;
4. Prune $C$;
5. Pass $C$ to the next phase;
6. $F$ is satisfiable iff there is some formula $f \in C$ which evaluates to True;
Correctness and time complexity of the algorithm:

- At the beginning of each stage, there is some satisfiable formula in $C$ iff at the end of this stage, there is some satisfiable formula in $C$.
- The size of $C$ at the beginning of each stage is at most $p_g(|F|) + 1$. 
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

**Sparse coNP-complete Language**

- Is there any sparse coNP-complete language?
- Sparse set $S$: there is a polynomial $p$ such that for all $n$, $|S^{\leq n}| \leq p(n)$. 

Sparse coNP-complete Language (Cont.)

Theorem

If there is a sparse set that is $\leq^p_m$-hard for coNP, then P=NP.
How to prove? Can we generalize the above method?

- $\text{SAT} \leq^p_m S^c$, that is there is a polynomial-time deterministic function $g$ many-one reducing $\text{SAT}$ to $S^c$.
- $S$ is a sparse set.
- $\text{SAT}$ is 2-disjunctive-self-reducible.
Motivation
Definitions
Tally NP-complete Languages
Sparse \( coNP \)-complete Language
Sparse NP-complete Language
Sparse languages in \( \text{NP} \)-\( \text{P} \)

Qi Ge and Xiaoming Gu
The Self-Reducibility Technique
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

\[ F[v_1=True] \]
\[ F[v_1=False] \]

It is satisfiable
Collide, no need
Keep it

\[ \Sigma^* \]
\[ S \]
\[ S_{\leq_p (|F|)} \]
Sparse coNP-complete Language (Cont.)

Algorithm

Input: $F$ with $v_1, v_2, \ldots, v_n$ as its variables

1. Initialization: $C = \{F\}$;
2. Do $i = 1$ to $n$
   3. Expand $C$;
   4. Prune $C$;
   5. If $C$ contains at least $p_d(p_k(|F|)) + 1$ elements, stop and declare $F \in SAT$;
   6. Pass $C$ to next phase;
   7. $F$ is satisfiable iff there is some formula $f \in C$ which evaluates to True;
Correctness and time complexity of the algorithm:

- At the beginning of each stage, there is some satisfiable formula in $C$ iff at the end of this stage, there is some satisfiable formula in $C$.
- The size of $C$ at the beginning of each stage is at most $p_d(p_k(|F|)) + 1$. 
Sparse NP-complete Language

Is there a sparse NP-complete language?
Theorem

If there is a sparse set that is \( \leq_{m}^{P} \)-complete for NP, then P=NP.
Sparse NP-complete Language (Cont.)

What do we have now?

- $S$ is a sparse set.
- $SAT \leq^p_m S$, that is there is a polynomial-time deterministic function $g$ many-one reducing $SAT$ to $S$.
- $SAT$ is 2-disjunctive-self-reducible.

What’s the problem?

- Is $SAT \leq^p_m S$ enough to get the result?
- Can we find a relationship between $S^c$ and $S$, e.g., $S^c \leq^p_m S$?
Definition

The census function $c_S$ of set $S$ is defined to be

$$c_S(n) = |\{x | x \in S \land |x| \leq n\}|.$$

For each sparse set $S$, we have that the census function of $S$ is bounded by some polynomial.
Sparse NP-complete Language (Cont.)

Why we choose polynomial-time computable census function?
- If the census function of some sparse NP-complete language $S$ is polynomial-time computable, then $S^c \in \text{NP}$.
- $\text{SAT} \leq^p_m S$ and $S^c \leq^p_m S \Rightarrow \text{SAT}^c \leq^p_m S$.
- Then we have $\text{P}=\text{NP}$ (by our previous result about sparse coNP-hard sets).
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

Theorem

If an NP-complete sparse language exists such that its census function $c_S$ is computable in polynomial time, then $P=NP$. 
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

Sparse NP-complete Language (Cont.)

NP Algorithm for $S^c$

Input: x

1. Let $n = |x|$, $k = c_S(n)$.
2. Guess $k$ distinct strings of length at most $n$.
3. Nondeterministicly test to make sure that all these $k$ strings are in $S$, and if they all are then accept iff $x$ is not one of these strings.

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
Next, we will prove the theorem that if $SAT^c \leq^p_m S$ for some sparse set $S$ then $P=NP$ via depth-first search and this approach will be useful in the later proof.
The Self-Reducibility Technique
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

The Self-Reducibility Technique

Qi Ge and Xiaoming Gu
Tally NP-complete Languages
Sparse coNP-complete Language
**Sparse NP-complete Language**
Sparse languages in NP-P

\[ F[v_1=True], F[v_1=False], F[v_1=True, \ldots, v_n=True] \]

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

Qi Ge and Xiaoming Gu
The Self-Reducibility Technique
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
**Sparse NP-complete Language**
Sparse languages in NP-P

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

F[\nu_1=\text{True}] F[\nu_1=\text{False}]

I don’t know, keep on going

f(\text{False})

\Sigma^*

Not satisfiable

Qi Ge and Xiaoming Gu
The Self-Reducibility Technique
Motivation

Definitions

Tally NP-complete Languages

Sparse coNP-complete Language

Sparse NP-complete Language

Sparse languages in NP-P

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
Algorithm for SAT

Input: formula $F$

1. Use a set $L$ to keep the images of unsatisfiable formula under $f$, initialize $L$ to be \{f(False)\};

2. If $F$ has no variables and evaluates to True, then $F$ is satisfiable;

3. If $f(F)$ is in $L$, then $F$ is unsatisfiable;

4. Otherwise recursively test $F[v_i = True]$ and $F[v_i = False]$;

5. If neither of the two formula is satisfiable then $F$ is unsatisfiable and add $f(F)$ to $L$.

6. Otherwise $F$ is satisfiable.
Correctness and time complexity of the algorithm:

- Tree pruning.
- The size of $L$ is at most $p(q(|F|))$.
- Visit at most $np(q(|F|)) + n - 1$ inner nodes.
What is the situation if the census function cannot be computed in polynomial time?

Just “guess” a value for census function.
Motivation
Definitions
Tally NP-complete Languages
Sparse coNP-complete Language
Sparse NP-complete Language
Sparse languages in NP-P

Sparse NP-complete Language (Cont.)

Define pseudo-complement of $S$ to be
\[
PC(S) = \{ \langle x, k, 0^n \rangle | |x| \leq n \land k \leq p(n) \land (k < c_S(n) \lor (k = c_S(n) \land x \in S^c)) \}.
\]

If $S$ is a sparse NP language, then $PC(S)$ is in NP.
Sparse NP-complete Language (Cont.)

NP Algorithm for $PC(S)$

Input: $\langle x, k, 0^n \rangle$

1. Check if $|x| \leq n \land k \leq p(n)$, if not, reject.
2. Guess $k$ distinct strings of length at most $n$.
3. Nondeterministically test to make sure that all these $k$ strings are in $S$, and if they all are then accept iff $x$ is not one of these strings.
What do we have now?

- \( SAT^c \leq^p_m S^c \)
- \( PC(S) \leq^p_m S \)
- \( S \) is sparse.
- \( SAT \) is 2-disjunctive-self-reducible.
**Motivation**

**Definitions**

**Tally NP-complete Languages**

**Sparse coNP-complete Language**

**Sparse NP-complete Language**

**Sparse languages in NP-P**

\[ F_n, k \]

\[ f_{n,k}(F) = g(h(F), k, 0^{p_{n}(n)}) \]

\[ h(F), k, 0^{p_{n}(n)} \]
If $k = c_S(p_h(|F|))$, $|\langle h(F), k, 0^{p_h(|F|)} \rangle|$ is bounded by some polynomial.

If $k = c_S(p_h(|F|))$, in the tree-traversal algorithm, at most some polynomial number of inner nodes will be visited.
Algorithm for SAT

Input: formula $F$

- For $k = 0$ to $p(p_h(|F|))$
- Use $f_{|F|,k}$ as pruning function to execute the tree-traversal algorithm and visit at most $|F|p(p_g(3|F|)) + |F| - 1$ inner nodes;
- If the algorithm accepts then accepts;
- reject;
Sparse NP-complete Languages (Cont.)

Correctness and time complexity of the algorithm:
- The algorithm accepts a formula $F$ iff for some $k$, there is a tree-traversal path to reach a leaf node which evaluates to True.
- The algorithm never rejects a formula which is satisfiable.
- The number of visited tree nodes is bounded by some polynomial.
E and NE

- $E = \bigcup_{c>0} \text{DTIME}[2^{cn}]$
- $\text{NE} = \bigcup_{c>0} \text{NTIME}[2^{cn}]$
- $E \subseteq \text{NE}$

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
Theorem

The following statements are equivalent
1. E=NE
2. NP-P contains no sparse sets
3. NP-P contains no tally sets
If NP-P contains no tally sets then E=NE.

Sub-Theorem (3⇒1)
If NP-P contains no tally sets then E=NE.

What we have:

- A. NP-P contains no tally sets
- B. P⊆NP
- C. E⊆NE
If NP-P contains no tally sets then $E = NE$.

Proof for $NE \subseteq E$

**Start Point**

$L \in NE \Rightarrow$ there exists a nondeterministic exponential-time TM $N_L$ and $L(N_L) = L$

**Construct a tally set**

$T = \{1^k | (\exists x \in L)[k = (1x)_{bin}]\}$
If NP-P contains no tally sets then E=NE.

\[ T \in \text{NP?} \text{ – YES.} \]

**Algorithm for \( T \)**
- Input string \( y \)
- Reject if \( y \) is not of the form \( 1^k \) \( (k > 0) \)
- Simulate \( N_L(x) \) \( (k = (1x)_{\text{bin}}) \) if \( y \) is of the form \( 1^k \) \( (k > 0) \)

The algorithm is nondeterministic and polynomial-time.

\( T \in \text{NP} \Rightarrow T \in \text{P} \Rightarrow \) there exists a deterministic polynomial-time TM \( M_T \) and \( L(M_T) = T \)
If NP-P contains no tally sets then $E = NE$.

**Algorithm for $L$**
- Input string $y$
- Get string $x = 1^{(1y)}_{bin}$
- Simulate $M_T(x)$ and accept if and only if $M_T(x)$ accepts

The algorithm is deterministic and exponential-time. So $L \in E$ and $NE \subseteq E$.
The claim is proved.
If $E=NE$ then $NP-P$ contains no tally sets.

Warm Up for Sub-Theorem($1 \Rightarrow 2$)
If $E=NE$ then $NP-P$ contains no sparse sets.

Sub-Theorem($1 \Rightarrow 3$)
If $E=NE$ then $NP-P$ contains no tally sets.

What we have:

- A. $E=NE$
- B. $P \subseteq NP$
If \( E = NE \) then \( NP-P \) contains no tally sets.

Proof for tally set \( T \in P \) if \( T \in NP \)

Start Point

Tally set \( T \in NP \Rightarrow \) there exists a nondeterministic polynomial-time TM \( N_T \) and \( L(N_T) = T \)

Construct an NE language

\[
L = \{ x \mid (x = 0 \lor x \text{ is a binary string of nonzero length with no leading zeros}) \land 1^{(x)_{bin}} \in T \}
\]
If E=NE then NP-P contains no tally sets.

$L \in \text{NE}$? – YES.

**Algorithm for $L$**

- Input string $y$
- Reject if $x \neq 0$ and $x$ is a binary string with leading zeros.
- Otherwise,
  - Get string $x = 1(y)_{\text{bin}}$.
  - Simulate $N_T(x)$ and accept if and only if $N_T(x)$ accepts.

The algorithm is nondeterministic and exponential-time.

$L \in \text{NE} \Rightarrow L \in \text{E} \Rightarrow$ there exists a deterministic exponential-time TM $M_L$ and $L(M_L) = L$
If $E = NE$ then NP-P contains no tally sets.

**Algorithm for $T$**

- Input string $y$
- Reject if $y$ is not of the form $1^k$ ($k \geq 0$)
- Otherwise, simulate $M_L(x)$ ($k = (x)_{\text{bin}}$) if $y$ is of the form $1^k$ ($k \geq 0$)

The algorithm is deterministic and polynomial-time. So $T \in P$ and no tally sets in NP-P.

The claim is proved.
If $E=NE$ then $NP-P$ contains no sparse sets.

Sub-Theorem ($1 \Rightarrow 2$)

If $E=NE$ then $NP-P$ contains no sparse sets.

What we have:

- A. $NE=E$
- B. $P \subseteq NP$
If \( E = NE \) then NP-P contains no sparse sets.

Proof for sparse sets \( S \in P \) if \( S \in NP \)

**Start Point**

Sparse set \( S \in NP \) \((\forall n)[||S=n|| \leq q(n)]\) \(\Rightarrow\) there exists a nondeterministic polynomial-time TM \( N_S \) and \( L(N_S) = S \)

**Difference from Warm Up**

How to construct an NE language \( L \) from \( S \) that a string in \( S \) can be checked by \( L \) easily?

Qi Ge and Xiaoming Gu

The Self-Reducibility Technique
If $E=NE$ then $NP-P$ contains no sparse sets.

Construct an NE language (HIS encoding set)

$$L = \{0\#n\#k \mid ||S^n|| \geq k\} \cup \{1\#n\#c\#i\#j \mid (\exists z_1, z_2, \ldots, z_c \in S^n)[z_1 <_{lex} z_2 <_{lex} \ldots <_{lex} z_c \land \text{the } j\text{th bit of } z_i \text{ is 1}]\}$$
If $E=NE$ then $NP-P$ contains no sparse sets.

$L \in NE? \quad – YES.$

**Algorithm for $L$**

- Input string $y$
- Reject if $y$ is syntactically illegal
- Check the first bit of $y$
  - The first bit of $y$ is 0.
    - Get binary value of $n$ and $k$.
    - Guess $k$ distinct strings that are $n$ bits long. If there is such a strings set that each string can be accepted by $N_S$ then accept. Otherwise, reject.
  - The first bit of $y$ is 1.
    - ...
If $E=NE$ then NP-P contains no sparse sets.

Algorithm for $L$ (Cont.)

- Input string $y$
- Reject if the string is not either form of $L$
- Check the first bit of $y$
  - The first bit of $y$ is 0.
    - ...
  - The first bit of $y$ is 1.
    - Get binary value of $n$, $c$, $i$ and $j$.
    - Guess $c$ distinct strings that are $n$ bits long. Then make them in lexical order. If there is such a set that each string is accepted by $N_S$ and the $j$th bit of $i$th string is 1 then accept. Otherwise, reject.
If $E = NE$ then $NP-P$ contains no sparse sets.

The algorithm is nondeterministic and exponential-time. $L \in NE \Rightarrow L \in E \Rightarrow$ there exists a deterministic exponential-time TM $M_L$ and $L(M_L) = L$. 
If $E=NE$ then NP-P contains no sparse sets.

**Algorithm for $S$**

- **Input string $y$**

- Let $n = |y|$ and simulate $M_L(0\#n\#0)$, $M_L(0\#n\#1)$, $M_L(0\#n\#2)$, $\ldots$, $M_L(0\#n\#q(n))$. Then get $c = \max\{k|0 \leq k \leq q(n) \land 0\#n\#k \in L\} = ||S^{=n}||$. 
If $E = NE$ then NP-P contains no sparse sets.

Algorithm for $S$ (Cont.)

- Simulate
  
  $M_L(1\#n\#c\#1\#1), ..., M_L(1\#n\#c\#1\#n),$
  
  $M_L(1\#n\#c\#2\#1), ..., M_L(1\#n\#c\#2\#n),$
  
  $..., M_L(1\#n\#c\#c\#1), ..., M_L(1\#n\#c\#c\#n).$

Then we can get all strings in $L$ that are $n$ bits long by checking each bit of them. So $y$ is accepted if and only if $y$ belongs to this set.

The algorithm is deterministic and polynomial-time. So $S \in P$ and no sparse sets in NP-P.

The claim is proved.
Summary

Two classes: E and NE.

Theorem 1.18 of Hem-Ogi: The three statements are equivalent.

- E=NE
- NP-P contains no sparse set
- NP-P contains no tally sets
Thanks!