Why Study Sparseness?

- Closely related to the p-isomorphism conjecture:

  \textbf{Lemma 5.1} Let $L_1$ and $L_2$ be two p-isomorphic languages. Then two polynomials $p_1$ and $p_2$ exist such that, for any $n$,

  \[ c_{L_1}(n) \leq c_{L_2}(p_1(n)) \quad \text{and} \quad c_{L_2}(n) \leq c_{L_1}(p_2(n)). \]

- In other words, the census functions are polynomially related.
- Since the census function for SAT is known to be exponential (Bov-Cre p. 83), if a sparse NP-complete language $L$ is found, then the census function for $L$ cannot be polynomially related to the census function for SAT. So, the p-isomorphism conjecture falls.
Difficulty of Finding a Sparse NP-C Language

• Due to a result by Mahaney:

\[\textbf{Theorem 5.7} \text{ If a sparse NP-complete language exists, then } P = NP.\]

• And thus, finding such an NP-complete language means proving that \(P = NP\), which is widely believed to be false.
• But why does Theorem 5.7 hold?
Table of Contents

- **Theorem 5.6**
  - Complement of S is in NP
  - Satisfiability tree pruning
  - SAT is in P

- **Theorem 5.7 (Mahaney’s Theorem)**
  - PC(S)
  - Constructing a good pruning function
  - A set of pruning functions
  - Properties of the correct pruning function
  - SAT is in P
Theorem 5.6 If an NP-complete sparse language exists such that its census function is computable in polynomial time, then $P = NP$. (A census function $c_L$ is said to be computable in polynomial time if a Turing transducer $T$ exists which reads $0^n$ as input and computes $c_L(n)$ in polynomial time.)
Table of Contents

- **Theorem 5.6**
  - Complement of S is in NP
  - Satisfiability tree pruning
  - SAT is in P

- **Theorem 5.7 (Mahaney’s Theorem)**
  - PC(S)
  - Constructing a good pruning function
  - A set of pruning functions
  - Properties of the correct pruning function
  - SAT is in P
Complement of S is in NP

Further, the complement of SAT reduces to S.
Table of Contents

● Theorem 5.6
   ○ Complement of S is in NP
   ○ Satisfiability tree pruning
   ○ SAT is in P

● Theorem 5.7 (Mahaney’s Theorem)
   ○ PC(S)
   ○ Constructing a good pruning function
   ○ A set of pruning functions
   ○ Properties of the correct pruning function
   ○ SAT is in P
Satisfiability Trees and Tree Pruning

\[
(x_1 \lor x_2 \lor \neg x_3) \land (\neg x_2) \land (\neg x_2 \lor x_3)
\]

\[
x_1 = t \quad x_1 = f
\]

\[
(\neg x_2) \land (\neg x_2 \lor x_3)
\]

\[
x_2 = t \quad x_2 = f
\]

\[
\text{false} \quad \text{true}
\]

\[
(x_2 \lor \neg x_3) \land (\neg x_2) \land (\neg x_2 \lor x_3)
\]

\[
x_2 = t \quad x_2 = f
\]

\[
\text{false} \quad (\neg x_3)
\]

\[
x_3 = t \quad x_3 = f
\]

\[
\text{false} \quad \text{true}
\]
Satisfiability Trees and Tree Pruning

- Since the complement of SAT reduces to S by some function f, we can use f as a pruning function.
- If y is a node in the satisfiability tree and f(y) is in S, then we know that y is not satisfiable, and we can ignore y’s children in the tree.
- It is difficult to determine if f(y) is in S directly since S is NP-C.
- To solve this, we build a list of elements in S as our tree pruning algorithm runs, which we can query.
Table of Contents

- **Theorem 5.6**
  - Complement of S is in NP
  - Satisfiability tree pruning
    - SAT is in P

- **Theorem 5.7 (Mahaney’s Theorem)**
  - PC(S)
  - Constructing a good pruning function
  - A set of pruning functions
  - Properties of the correct pruning function
  - SAT is in P
SAT is in P

begin {input: x} {main program}
  list := \{f(false)\};
  if sat(x) then accept else reject;
end.

function sat(y): Boolean;
begin
  if y = true then sat := true;
  if f(y) \in list then sat := false
  else
    begin
      derive y_0 and y_1 from y;
      if \neg sat(y_0) \land \neg sat(y_1) then
        begin
          {if both y_0 and y_1 are not satisfiable then y is not satisfiable} 
          list := list \cup \{f(y)\};
          sat := false;
        end
      else sat := true;
    end
end;
Mahaney’s Theorem

**Theorem 5.7** If a sparse NP-complete language exists, then $P = NP$. 
Table of Contents

- **Theorem 5.6**
  - Complement of S is in NP
  - Satisfiability tree pruning
  - SAT is in P

- **Theorem 5.7 (Mahaney’s Theorem)**
  - PC(S)
  - Constructing a good pruning function
  - A set of pruning functions
  - Properties of the correct pruning function
  - SAT is in P
Mahaney’s Theorem

- We have shown that we can prove $P = NP$ if the sparse NP-C set has a poly-time computable census function.
- Can we still do it if we don’t have this assumption?
- Yes, and it will require “guessing” the correct value of the census function.
Pseudo-complement of S: (PC(S))

Define PC(S) as the set of triples $<x,k,0^n>$ accepted by this machine:

```
begin {input: x, k, 0^n}
  if $|x| > n \lor k > p(n)$ then reject;
  guess $y_1, \ldots, y_k$ in set of $k$-tuples of distinct words
  each of which is of length, at most, $n$;
  for $i = 1$ to $k$ do
    if $NT(y_i)$ rejects then reject;
  for $i = 1$ to $k$ do
    if $y_i = x$ then reject;
  accept;
end.
```

Note that if $|x| \leq n$ and $k = c_s(n)$, then $<x,k,0^n>$ is in the set if and only if $x$ is in $S$ complement.
# Table of Contents

- **Theorem 5.6**
  - Complement of $S$ is in NP
  - Satisfiability tree pruning
  - SAT is in P

- **Theorem 5.7 (Mahaney’s Theorem)**
  - $PC(S)$
  - Constructing a good pruning function
  - A set of pruning functions
  - Properties of the correct pruning function
  - SAT is in P
Constructing a good pruning function

- Before, we used a reduction from the complement of SAT to S. This time we do the exact same thing, with a new reduction.
- First, let $h$ be a reduction from SAT to S, and $g$ be a reduction from $PC(S)$ to S.
- Let $p_h$ and $p_g$ limit the length of $h$ and $g$.
- Now, suppose that $x$, the string we want to determine is satisfiable or not, is of length $n$.
- Define $F^*(y) = g(h(y), c_s(p_h(n)), 0^{p_h(n)})$
- Claim: $F^*(y)$ is a reduction from the complement of SAT to S.
Table of Contents

- **Theorem 5.6**
  - Complement of S is in NP
  - Satisfiability tree pruning
  - SAT is in P

- **Theorem 5.7 (Mahaney’s Theorem)**
  - PC(S)
  - Constructing a good pruning function
  - A set of pruning functions
  - Properties of the correct pruning function
  - SAT is in P
A set of pruning functions

- For each \( n \) and for each \( k \leq p(p_h(n)) \), where \( p \) is a polynomial bounding the census function for \( S \):
- Define
  \[
  f_{n,k}(y) = g(h(y), k, 0^{p_h(n)}).
  \]
- These are all poly-time computable when \( |y| \leq n \)
- Note that \( F^*(y) = g(h(y), c_s(p_h(n)), 0^{p_h(n)}) = f_{n,c_s(p_h(n))}(y) \)
- So, \( F^* \) is among this set of functions, and it is a poly-time reduction from the complement of SAT to \( S \), thus it is a valid pruning function.
- The problem: we do not know the census function, so we cannot compute \( F^*(y) \).
A set of pruning functions

- Observation: $c_s(p_h(n)) \leq p(p_h(n))$
- Solution idea: On input x of size n, for all values of k from 0 to $p(p_h(n))$, try the tree-pruning algorithm with pruning function $f_{n,k}$
- One of these k’s will work.
- Problem: we are not guaranteed that the tree-pruning algorithm will run in poly time with every choice of pruning function.
Table of Contents

- **Theorem 5.6**
  - Complement of S is in NP
  - Satisfiability tree pruning
  - SAT is in P

- **Theorem 5.7 (Mahaney’s Theorem)**
  - PC(S)
  - Constructing a good pruning function
  - A set of pruning functions
  - Properties of the correct pruning function
  - SAT is in P
Properties of the correct pruning function $F^*$

- Observation: if $k = c_s(p_h(n))$, we have that
  \[ |\langle h(y), k, 0^{p_h(n)}\rangle| = 2p_h(n) + O[\log(p(p_h(n)))]| \]

- Thus, there is an $n_0$ such that for all $n \geq n_0$,
  \[ |\langle h(y), k, 0^{p_h(n)}\rangle| \leq 2p_h(n) + p(p_h(n)) \]

- This means that the list created by the tree-pruning algorithm will be at most
  \[ p(p_g[2p_h(n) + p(p_h(n))]) \]
Properties of the correct pruning function $F^*$

- So, we know from the poly-bound proof of the tree-pruning function, that the algorithm visits at most

$$2[|x| p(p_g[2p_h(n) + p(p_h(n))]) + |x| - 1]$$

Nodes, which is polynomial in $|x|$.

- Further, if the amount of nodes explored exceeds this value, we know we have chosen the wrong $k$. 

Table of Contents

- Theorem 5.6
  - Complement of S is in NP
  - Satisfiability tree pruning
  - SAT is in P

- Theorem 5.7 (Mahaney’s Theorem)
  - PC(S)
  - Constructing a good pruning function
  - A set of pruning functions
  - Properties of the correct pruning function
  - SAT is in P
begin \{ input: \( x \) \}
    \begin{align*}
    &\text{for } k = 0 \text{ to } p(p_h(|x|)) \text{ do} \\
    &\quad \text{begin} \\
    &\quad \quad \text{execute the tree-visiting algorithm described in the} \\
    &\quad \quad \text{proof of Theorem 5.6 using } f_{|x|,k} \text{ as a pruning function} \\
    &\quad \quad \text{and visiting, at most, } 2[|x|p(p_g[2p_h(n) + p(p_h(n))]) + |x| - 1] \text{ inner nodes;} \\
    &\quad \quad \text{if the algorithm accepts then accept;} \\
    &\quad \text{end;} \\
    &\text{reject;} \\
    &\text{end.}
    \end{align*}