Improving Known Solutions is Hard

Desh Ranjan*
Suresh Chari
Pankaj Rohatgi**

TR 90-1171
November 1990

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Supported by an MSI Fellowship.
**Supported by NSF Research Grant CCR-88-23053.

Improving known solutions is hard

Desh Ranjan [*] Suresh Chari

Pankaj Rohatgi [†]

Computer Science Department

Cornell University

November 1990

Abstract

In this paper, we study the complexity of computing better solutions to optimization problems given other solutions. This is done in the context of the counterexample computation model introduced in [KPS90]. Assuming $PH \neq \Sigma_3^P$, we prove that $PTIME$ transducers cannot compute optimal solutions for many problems, even given $O(n^{1-\epsilon})$ *non-trivial* solutions. These results are used to establish sharp lower bounds for several problems in the counterexample model. We extend the model by defining probabilistic counterexample computations and show that our results hold even in the presence of randomness.

1 Introduction

Efficient solution of optimization problems is one of the most challenging tasks in computer science. In particular, the characterization of the difficulty of computing optimum solutions of NP-optimization problems is of great interest and importance in practice and theory.

Several models have been proposed to study NP-optimization problems, though there is no universally accepted "best" one [AMSP80,Kre88,PY88]. In a recent investigation Krajíček *et al* [KPS90]have defined a new model for studying the difficulty of optimization problems called the "counterexample" model. In this model there is an all-powerful teacher, T, and a student, S, with limited power($PTIME$). The goal of S is to compute the optimum solution. To this end she is aided by T in the following way : at any point in the computation S may present a solution claiming it to be optimal. If there is no better solution

[*]Supported by MSI Fellowship

[†]Supported by NSF Research Grant CCR-88-23053

T accepts the claim, else T disproves the claim by presenting a counterexample, *i.e.* a better solution. The difficulty of a problem is measured by the number of counterexamples the best student requires to compute the optimum solution given the least cooperative teacher. The important questions in this model concern the difficulty of various optimization problems. The model is also interesting because it relates conjectures about the relative powers of logical theories [KPS90] to those about this model which are purely computational in nature.

In the next section we briefly review the counterexample model, giving the necessary definitions and relevant results. In Section 3, we establish the difficulty of computing the lexicographically maximum satisfying assignment of boolean formulas in this model. The proof uses a result(Lemma 1) which is central to this paper, and interesting in its own right. It states that, under the assumption that PH does not collapse, not only is it hard to find satisfying assignments for satisfiable formulas but there are formulas for which it is hard to find a *new* satisfying assignment even given a lot of *non-trivial* ones. We use similar results to prove sharp bounds for the computation of the optimum solution for several graph-theoretic NP-optimization problems such as $MAXCLIQUE$. We then define probabilistic counterexample protocols and prove that these bounds exist even if the student has access to the power of randomness.

2 Preliminaries

In this section we review the basic definition of the computation model and summarize known results. The definitions are variations of those used in [KPS90].

Definition 1 *An NP-optimization problem is binary relation $R \subseteq \Sigma^* \times \Sigma^*$, a function $\sigma : \Sigma^* \times \Sigma^* \rightarrow N$ and a polynomial p such that R, σ are computable in polynomial time. y is called a* feasible solution *of x if $|\, y \,| \leq p(|\, x \,|)$ and $R(x, y)$. Given two feasible solutions y and y' of x, we say that y' is a better solution than y if $\sigma(x, y') > \sigma(x, y)$. A feasible solution y is optimal if there is no other feasible y' which is better than y.*

Example: The NP-optimization problem $MAXCLIQUE$ can be defined by $R(G, y)$="*y is a clique in the graph G*" and $\sigma(G, y)$= number of vertices in y.

Example: The canonical NP-optimization problem $LEXMAXSAT$ is defined by $Q(F, s)$ ="*s is a satisfying assignment of F*" and $\rho(F, s) = s$. Here we assume that s, a truth assignment of F, is an n-bit string where n is the number of variables in F and $s_i = 1$ iff the i-th variable is assigned *true*.

2.1 The Counterexample model

A S-T *counterexample protocol* for an optimization problem consists of a deterministic polynomial time machine S, called the student and an all powerful Turing machine T, the teacher. Given an instance of the problem the goal of the student is to produce an optimum solution. During the computation the student repeatedly presents the teacher with feasible solutions, claiming them to be optimal. The teacher either produces a better solution, *i.e.* a *counterexample*, or accepts if there is none. The computation ends when the teacher accepts. Note that the student is restricted to spending polynomial time before producing a new feasible solution. However she is allowed an arbitrary number of steps over the entire computation.

Definition 2 *An NP-optimization problem $\mathcal{P}$ has an $f(n)$-counterexample protocol if there is a student S such that for all teachers T, S-T forms a counterexample protocol for $\mathcal{P}$ which requires no more than $f(n)$ counterexamples on inputs of size n.*

Definition 3 *$\mathcal{C}[f(n)]$ is the class of all NP-optimization problems which have an $f(n)$-counterexample protocol.*

It is easy to see that $MAXCLIQUE$ has a n-counterexample protocol. Given a graph G the student first presents a one vertex clique. Then the student repeatedly presents the same solution the teacher provides as a counterexample. We call such a strategy the *trivial strategy*. Clearly in this case the strategy takes at most n counterexamples. However it is not clear that there is *poly*-counterexample protocol for $LEXMAXSAT$. As stated previously the interesting questions in this model relate to the number of counterexamples required to compute the optimum solution of a given problem. It is established in [KPS90] that

Theorem 1 *$\forall f$ if $1 \leq f(n) \leq n^{1-\epsilon}$, $\epsilon > 0$, and f is polynomial time constructible then $\mathcal{C}[f(n)] = \mathcal{C}[f(n)-1]$ implies $NP \subseteq P/poly$.*

This shows that extra counterexamples help. As an initial step towards characterizing the difficulty of solving NP-optimization problems using the counterexamples the following theorem is also proved

Theorem 2 *If PH does not collapse then* there exists *an $\epsilon > 0$ such that there is no $n^{1-\epsilon}$-counterexample protocol for $LEXMAXSAT$ and $MAXCLIQUE$.*

In the following sections we extend the above result, showing that for *for all $\epsilon > 0$* there is no $n^{1-\epsilon}$-counterexample protocol for these problems. Since $MAXCLIQUE$ has a n-counterexample protocol, this gives a tight bound on the number of counterexamples required for computing a clique of maximum size in a graph. We use our techniques to prove such bounds for many other problems. In fact, we show the same result for these problems even in presence of randomness.

3 The Complexity of Computing New Satisfying Assignments

We know that if $P \neq NP$ then it is difficult to compute the satisfying assignment of boolean formulas. The question remains whether there are formulas for which it is hard to compute a new satisfying assignment given a lot of other satisfying assignments. We answer this question positively, under the assumption that PH does not collapse.

In the following lemma we use the language $USAT$, the set of boolean formulas with a unique satisfying assignment. It is known that if $USAT \in coNP/poly$ then PH collapses. For a short proof see the appendix.

Notation: For $F \in USAT$, $s(F)$ denotes the unique satisfying assignment of F. For any set A, $A^{=n}$ denotes $\{x \in A : |x| = n\}$.

Lemma 1 *Assume $USAT \notin coNP/poly$. Let D be any deterministic polynomial time transducer and $r(n)$ be any polynomial. Then for infinitely many n, there are boolean formulas $F_1, F_2, \ldots, F_{r(n)} \in USAT^{=n}$ such that for all j, $1 \leq j \leq r(n)$, D given $\{F_1, \ldots, F_{r(n)}, s(F_1), \ldots, s(F_{j-1}), s(F_{j+1}), \ldots, s(F_{r(n)})\}$ can not compute $s(F_j)$.*

Proof: Suppose, to the contrary, there exists a deterministic polynomial time transducer D and a polynomial $r(n)$ such that for all but finitely many n the above fails. This means that given any set $F=\{F_1, F_2, \ldots, F_{r(n)}\}$ of formulas in $USAT^{=n}$ the following condition holds:

- D given $\{F_1, F_2, \ldots, F_{r(n)}, s(F_2), \ldots, s(F_{r(n)})\}$ computes $s(F_1)$, or
- D given $\{F_1, F_2, \ldots, F_{r(n)}, s(F_1), s(F_3), \ldots, s(F_{r(n)})\}$ computes $s(F_2)$, or

  $\vdots$
- D given $\{F_1, F_2, \ldots, F_{r(n)}, s(F_1), \ldots, s(F_{r(n)-1})\}$ computes $s(F_{r(n)})$.

We use this to find a set of $r(n) - 1$ formulas in $USAT^{=n}$ knowing whose satisfying assignments D can compute the unique satisfying assignment for a substantial fraction of the formulas in $USAT^{=n}$. Let $Z \subseteq USAT^{=n}$. For each set $\mathcal{G}=\{G_1, \ldots G_{r(n)-1}\} \subseteq Z$ define

$$R^Z_{\mathcal{G}} = \{H \in Z \mid D \text{ given } G_1, \ldots, G_{r(n)-1}, H, s(G_1), \ldots, s(G_{r(n)-1}) \\ \text{computes } s(H)\}.$$

$R^Z_{\mathcal{G}}$ is the set of all formulas in Z whose satisfying assignment can be computed by D given the formulas in $\mathcal{G}$ and their satisfying assignments. We claim that for some set $\mathcal{A} \subseteq Z$ of $r(n) - 1$ formulas, $|R^Z_{\mathcal{A}}| \geq (|Z| - r(n))/r(n)$.

To see why this is true, consider any set $\mathcal{G} = \{G_1, G_2, \ldots, G_{r(n)}\} \subseteq Z$. Since the above condition holds, there is an i, $1 \leq i \leq r(n)$, such that D given $\{G_1, G_2, \ldots, G_{r(n)}, s(G_1), \ldots, s(G_{i-1}), s(G_{i+1}), \ldots, s(G_{r(n)})\}$ computes $s(G_i)$.

Hence $G_i \in R_{\mathcal{G}\setminus\{G_i\}}$. Hence for every set $\mathcal{G}'$, of size k, there is a formula $H \in \mathcal{G}'$ such that $H \in R^Z_{\mathcal{G}'\setminus H}$. Therefore

$$\begin{aligned}\textstyle\sum_{\mathcal{B}\subseteq Z, |\mathcal{B}|=r(n)-1} \mid R^Z_{\mathcal{B}} \mid &\geq \#(r(n)\text{-element subsets of } Z)\\ &= \binom{\mid Z \mid}{r(n)}.\end{aligned}$$

Thus there is some set $\mathcal{F} = \{F_1, \ldots, F_{r(n)}\} \subseteq Z$ such that,

$$\mid R^Z_{\mathcal{F}} \mid \geq \binom{\mid Z \mid}{r(n)} / \binom{\mid Z \mid}{r(n)-1} = (\mid Z \mid - r(n))/r(n).$$

This implies that given $F_1, \ldots, F_{r(n)-1}$ and their satisfying assignments, D can compute the satisfying assignments for about $1/r(n)$ of the formulas in Z.

This needs to be repeated at most $nr(n)$ times to obtain a polynomially long advice, given which, we can compute satisfying assignments for all formulas in $USAT^{=n}$ in polynomial time. The algorithm in Fig. 1 constructs this advice. Given this advice $A = A_1\#A_2\#\ldots\#A_k$, a $coNP$ machine which works as follows can recognize $USAT$.

Given boolean formula F, of length n, as input

- Check that F has no more than one satisfying assignment.
- **if** F and its satisfying assignment appears in the advice , **or** for some j, D given A_j computes $s(F)$ **then** $ACCEPT$ **else** $REJECT$.

Hence, $USAT \in coNP/poly$ contradicting the assumption. This completes the proof of the lemma. ∎

Technicalities apart, the lemma says that there are large collections of equal sized boolean formulas whose satisfying assignments are "orthogonal", in the sense that satisfying assignments of one provide no information about those of any other. It is then possible to construct boolean formulas, with several satisfying assignments, for which the different satisfying assignments are orthogonal in the above sense. We use the existence of such formulas to improve the lower bounds for the number of counterexamples required for $LEXMAXSAT$.

Theorem 3 *Unless the Polynomial Hierarchy collapses to* Σ^P_3, for all $\epsilon > 0$, $LEXMAXSAT \notin \mathcal{C}[n^{1-\epsilon}]$.

Proof: Choose any $\epsilon > 0$. Assume that there is an $n^{1-\epsilon}$-counterexample protocol for $LEXMAXSAT$. Let D be the student in this protocol. Let $r(n)$ be a polynomial such that $(nr(n))^{1-\epsilon} < r(n)$. Define polynomial time transducer D' which on input $\{H_1, H_2, \ldots H_{r(n)}, y_1, \ldots y_{k-1}, y_{k+1} \ldots y_{r(n)}\}$ does the following:

- Checks each y_i is a satisfying assignment of H_i.

```
$Z \leftarrow USAT^{=n}$
$A \leftarrow$ "null string" /* $A$ is the advice string to be computed */
While $|Z| \geq r(n)$ do
    begin
        find $\mathcal{B} \subseteq Z$ of size $r(n)-1$ such that $R_{\mathcal{B}}^{Z} \geq |Z|/r(n)$
        Let $\mathcal{B}=\{B_1, B_2, \ldots B_{r(n)-1}\}$
        /* One such B exists by the proof */
        $Z \leftarrow Z \setminus R_{\mathcal{B}}^{Z}$
        $A \leftarrow A\#\mathcal{B}, s(B_1), s(B_2), \ldots s(B_{r(n)-1})\#$
    end

$A \leftarrow A\#Z, s(Z_1), s(Z_2) \ldots s(Z_m)\#$
/* where $Z = \{Z_1, Z_2 \ldots Z_m\}$ */
```

Figure 1: Algorithm to construct advice for strings of length n

- D' simulates D on $H = H_1 \bigvee H_2 \ldots \bigvee H_{r(n)}$. Whenever D presents a solution y to the teacher, if y is a satisfying assignment of H_k then D' outputs y and stops. If on the other hand $y = y_i$, D' continues the simulation assuming the teacher provided the lexicographically next satisfying assignment from the y_j's. In all other cases D' outputs some arbitrary fixed string.

If PH does not collapse then, by Lemma 1, there are infinitely many n such that there are $r(n)$ formulas $F_1, F_2, \ldots F_{r(n)} \in USAT$ such that D' given satisfying assignments for any $r(n)-1$ of these cannot compute the satisfying assignment for the remaining one. Consider the formula $F = F_1 \bigvee F_2 \ldots \bigvee F_{r(n)}$ where the variables of all the F_i's are same. Then the **only** satisfying assignments of F are $s(F_1)$,$s(F_2)$,..., $s(F_{r(n)})$. Moreover for all i, D', given F as input, cannot compute $s(F_i)$ even given all the remaining $s(F_j)$'s. Assume that the teacher provides the satisfying assignments of F in lexicographical order. Since the length of F is $c.nr(n)$ for some constant c, D can ask for at most $(c.nr(n))^{1-\epsilon}$ counterexamples. By choice of r this is less that $r(n)$ for large n. This means that D cannot follow the trivial strategy. Hence at some point of time in the protocol D computed a *new* satisfying assignment for F by itself. Suppose that the first time this happened, $s(F_i)$ was produced. Then, by construction of D', $D'(F_1, F_2, \ldots, F_{r(n)}, s(F_1), \ldots, s(F_{i-1}), s(F_{i+1}), \ldots, s(F_{r(n)}))$ outputs $s(F_i)$ which is a contradiction.

Note that D requires at least $r(n)$ counterexamples where n is the number of variables of F. Thus, for all polynomials p there is no $p(n)$-counterexample protocol where n is the number of variables. ∎

Corollary 1 *$\forall \epsilon > 0$, in any counterexample protocol for $LEXMAXSAT$, the student can be forced to follow the trivial strategy for $O(n^{1-\epsilon})$ steps on infinitely many formulas.*

We now consider the NP-optimization problem $MINTSP$ defined by the relation $R(G,t) =$" *t is a tour in the weighted graph G*" and $\rho(G,t)$ = length of the tour t.
In order to establish lower bounds for $MINTSP$ we use the language

$$UNIQOPTSP = \{G \mid G \text{ has a unique tour of minimum size}\}$$

$UNIQOPTSP$ is known to be NP-hard[Pap84]. Thus if $UNIQOPTSP \in CoNP/poly$ then PH collapses. As in the case of $LEXMAXSAT$ we establish the following lemma to prove the result.
Notation: For any graph G in $UNIQOPTSP$, $t(G)$ denotes its unique minimum tour.

Lemma 2 *Assume $UNIQOPTSP \notin coNP/poly$. Let D be any deterministic polynomial time transducer and $r(n)$ any polynomial. Then for infinitely many n, there are $G_1, G_2, \ldots, G_{r(n)} \in UNIQOPTSP^{=n}$, such that for all j, $1 \leq j \leq r(n)$, D given $\{ G_1, \ldots, G_{r(n)}, t(G_1), \ldots, t(G_{j-1}), t(G_{j+1}), \ldots, t(G_{r(n)}) \}$ can not compute $t(G_j)$.*

Proof: Similar to the proof of Lemma 1.
Using this we establish the desired lower bound on the number of counterexamples required to compute MINTSP.

Theorem 4 *Unless the Polynomial Hierarchy collapses to Σ_3^P,* for all $\epsilon > 0$, *$MINTSP \notin \mathcal{C}[n^{1-\epsilon}]$.*

Proof: The proof is similar to that of Theorem 3. Assume that there is an counterexample protocol for $MINTSP$ which requires at most $n^{1-\epsilon}$ counterexamples. Let D be the student in this protocol. Let $r(n)$ be a polynomial such that $(nr(n))^{1-\epsilon} < r(n)$. Define transducer D' as follows:
D' on input $\{G_1, G_2, \ldots G_{r(n)}, t(G_1) \ldots, t(G_{k-1}), t(G_{k+1}) \ldots t(G_{r(n)})\}$

- Checks each $t(G_i)$ is a tour of G_i.
- D' now constructs a new graph G from the input graphs $G_1, \ldots, G_{r(n)}$. Conceptually the new graph G can be thought of as a *chain* G_1–G_2– –G_n. The link between G_1 and G_2 is constructed as follows:
Let $u \in G_1$ and $v \in G_2$ be two vertices which are not part of any other link. Make two copies of u (say u' and u'') in G_1. Link u' and u'' to all the vertices to which u was linked keeping the edge weights same. Also link u' and u'' with a 0 weight edge. Do the same for v. Finally add the

0 weight edges (u', v') and (u'', v''). The remaining graphs are linked in a similar fashion. Clearly $|G| \leq cnr(n)$ for some constant c. By construction every tour of G provides tours for each of the graphs G_i. Moreover given tours for $G_1, \ldots, G_{r(n)}$, one can build a tour for G. It is not hard to show that if all the G_i's are in $UNIQOPTSP$ then so is G. In that case $t(G) = t(G_1) + t(G_2) + \ldots + t(G_{r(n)}$.

D' then simulates D on G. Whenever D presents a solution S to the teacher, D' finds the smallest $j \neq k$ (if any), such that the tour of G_j provided S is not minimum. D' then constructs a better solution S' by replacing the tour of G_j in S by the optimal tour $t(G_j)$. D' then continues the simulation assuming that the teacher T provided S' as a counterexample. If D' cannot improve S in this way (i.e, it can't find such a j), it then outputs the minimal tour of G_k which was ever output by the student.

Now exactly as in Theorem 3, if PH does not collapse, by Lemma 2 there are infinitely many n and graphs $G_1, G_2, \ldots, G_{r(n)} \in UNIQOPTSP$ such that D' given these graphs and the solutions to all but one of these, cannot compute the minimal tour of the remaining graph. This contradicts the assumption that D works within the claimed bound. ∎

4 Lower Bounds in Counterexample Computations

In the previous section we showed that if PH is infinite then $LEXMAXSAT$ and $MINTSP$ do not have $(n^{1-\epsilon})$-counterexample protocols. However, it is not known whether these problems even have polynomial-counterexample protocols. In this section we consider some NP-optimization problems which have n-counterexample protocols and establish strong lower bounds on the number of counterexamples required in any protocol to compute their optimal solutions. In particular we consider $MAXCLIQUE$ and the following optimization problems

$MAXINDSET$: The NP-optimization problem defined by the relation $R(G,y)=$ " *y is an independent set of vertices in G*" and $\sigma(G,y)=$ the number of vertices in y.

$MINCOVER$: defined by the relation $(G, y) =$" *y is a vertex cover in the graph G*" and $\rho(G,y)= n - | y |$ where n is the number of vertices of G.

In order to establish bounds for $MAXINDSET$ we use the language

$UNIQ\_INDSET = \{G \mid G$ has a unique independent set of maximum size$\}$

The reduction from $3CNFSAT$ to $MAXINDSET$ given in [GJ79] can be modified to yield a reduction from $USAT$ to $UNIQ\_INDSET$. This implies that if $UNIQ\_INDSET \in coNP/poly$ then $USAT \in coNP/poly$ and hence by Lemma 5 PH collapses. As in the case of $LEXMAXSAT$ we establish the following lemma to prove the result.

Notation: For any graph G in $UNIQ\_INDSET$, $i(G)$ denotes its unique maximum independent set. Also, $UNIQ\_INDSET^{=n}$ denotes the set of all n-vertex graphs in $UNIQ\_INDSET$.

Lemma 3 *Assume $UNIQ\_INDSET \notin coNP/poly$. Let D be any deterministic polynomial time transducer and $r(n)$ any polynomial. Then for infinitely many n, there are $G_1, G_2, \ldots, G_{r(n)} \in UNIQ\_INDSET^{=n}$, such that for all j, $1 \leq j \leq r(n)$, D given $\{ G_1, \ldots, G_{r(n)}, i(G_1), \ldots, i(G_{j-1}), i(G_{j+1}), \ldots, i(G_{r(n)}) \}$ and the size of $i(G_j)$ can not compute $i(G_j)$.*

Proof: Suppose the hypothesis is false. As in Lemma 1, there is a polynomial time machine D which, given a polynomially long advice $S = S_1 \# S_2 \ldots \# S_k$ and $d = |i(G)|$ can compute $i(G)$, for all $G \in UNIQ\_INDSET^{=n}$. Here each S_i is the encoding of $r(n)-1$ n-vertex graphs and their unique maximum independent sets.

Using this advice a $coNP$ machine which works as follows can recognize $UNIQ\_INDSET$.
Given a graph G with n vertices as input

- For each value of d from 1 to n run D on G,S and d.Then for some value of d, D computes an independent set of largest size l.
- Check that there is no more than one independent set of size l in G.

This contradicts the assumption that $UNIQ\_INDSET \notin coNP/poly$. ∎

Using this we establish the desired lower bound on the number of counterexamples required to compute the maximum independent set of a graph.

Theorem 5 *Unless PH collapses,* for all $\epsilon > 0$ *any protocol for $MAXINDSET$ requires at least $O(n^{1-\epsilon})$ counterexamples on infinitely many n-vertex graphs.*

Proof: The proof is similar to that of Theorem 3. Assume that there is an counterexample protocol for $MAXINDSET$ which requires at most $n^{1-\epsilon}$ counterexamples on n-vertex graphs. Let D be the student in this protocol. Let $r(n)$ be a polynomial such that $(nr(n))^{1-\epsilon} < r(n)$. Define transducer D' as follows:
D' on input $\{G_1, G_2, \ldots, G_{r(n)}, y_1, \ldots, S_{t-1}, S_{t+1}, \ldots, S_{r(n)}, k\}$

- Checks each S_i is an independent set of vertices in G_i.
- D' simulates D on G, the disjoint union of $G_1, G_2, \ldots, G_{r(n)}$. Whenever D presents a solution S to the teacher, if S contains at least k vertices of G_t then it outputs these vertices and stops. Otherwise it finds the smallest j, such that all vertices of S_j do not appear in S, and replaces the vertices of G_j in S by S_j and continues the simulation. In all other cases it outputs some fixed string.

Now exactly as in Theorem 3, if PH does not collapse, by Lemma 3 there are infinitely many n and graphs $G_1, G_2, \ldots, G_{r(n)} \in UNIQ\_INDSET$ such that D on input G, the disjoint union of $G_1, G_2, \ldots, G_{r(n)}$ cannot compute the maximum independent set of G, contradicting the assumption. ∎

Using the same method we can show that other optimization problems such as $MAXCLIQUE$, $MINCOVER$, $MAXCYCLE$ require at least $n^{1-\epsilon}$ counterexamples *for all* $\epsilon > 0$, on graphs of n vertices. These problems can be solved using by an n-counterexample protocol in which the student adopts the trivial strategy. It is startling that the this strategy is essentially the best (possibly modulo *polylog* factors).

5 Protocols with Randomness

In this section we consider probabilistic *counterexample protocols* in which the student is allowed access to a random source such as a fair coin. We will show that in many cases this additional power does not help the student substantially.

Definition 4 *An NP-optimization problem Q has a probabilistic(or random) $f(n)$-counterexample protocol if there is a student, S, and a polynomial q_S, such that for all T, S-T forms a probabilistic counterexample protocol for Q. In addition, with probability $\geq 1/q_S(n)$, this protocol should require no more than $f(n)$-counterexamples on inputs of size n.*

In the above definition, the probability is taken over the strings z of size $s(n)$ which are provided as input to S uniformly at random. Here $s(n)$ is some polynomial.

Definition 5 *$\mathcal{PC}[f(n)]$ is the class of all NP-optimization problems which have a probabilistic $f(n)$-counterexample protocol.*

Notice that the student is required to work within the counterexample bound with a probability which can be as low as $1/$*poly*. For any NP-optimization problem, there is always an $O(1)$ probabilistic protocol which works with probability $1/$*exponential*. In this section we will show that if the student is required to work with "significantly" better probability, then there are are NP-optimization problems which do not have $n^{1-\epsilon}$-counterexample protocols for any $\epsilon > 0$, unless the PH collapses. We first show that $LEXMAXSAT$ is one such problem.

Lemma 4 *Suppose $USAT \notin BP \cdot (coNP/poly)$. Let P be any probabilistic polynomial time transducer and $r(n)$ and $p(n)$ be any polynomials. Then for infinitely many n, there are formulas $F_1, F_2, \ldots, F_{r(n)} \in USAT^{=n}$ such that $\forall j$, $1 \leq j \leq r(n)$,D given $F_1, \ldots, F_{r(n)}, s(F_1), \ldots, s(F_{j-1})$, $s(F_{j+1}), \ldots, s(F_{r(n)})$, cannot compute $s(F_j)$ with probability $\geq 1/p(n)$.*

Proof: Generalization of the proof of Lemma 1.

Theorem 6 *Unless the PH collapses, $\forall \epsilon > 0$, $LEXMAXSAT \notin \mathcal{PC}[n^{1-\epsilon}]$.*

Proof: Assume that the Polynomial Hierarchy is infinite and there is a probabilistic $n^{1-\epsilon}$-counterexample protocol for $LEXMAXSAT$. Let P be the student in this protocol which works with probability $1/t(n)$. Define a transducer P' exactly as in Theorem 3. Let $q(n) = t(r(n) * n^2)$. From Lemma 4, for $p(n) = q(n) * r(n) * n$ there are infinitely many n's such that there are $r(n)$ formulae $F_1, F_2, \ldots F_{r(n)} \in USAT^{=n}$ such that P' given satisfying assignments for any $r(n) - 1$ of these cannot compute the satisfying assignment for the remaining one with probability $\geq 1/p(n)$. As in Theorem 3, we consider the formula $F = F_1 \bigvee F_2 \ldots \bigvee F_{r(n)}$ where the variables of all the F_i's are same. Let z be the random input to P and let $|z| = s(n)$ for some polynomial s. For every random input z to P, such that P does not follow the trivial strategy, there is an i, $1 \leq i \leq r(n)$ such that with random input z, $P'(F_1, F_2, \ldots, F_{r(n)}, s(F_1), \ldots, s(F_{i-1}), s(F_{i+1}), \ldots, s(F_{r(n)})$ outputs $s(F_i)$. By definition, more than $1/q(n)$ of all strings z of size $s(n)$ cause P to avoid the trivial strategy. A simple counting argument shows that there is a j such that $P'(F_1, F_2, \ldots, F_{r(n)}, s(F_1), \ldots, s(F_{j-1}), s(F_{j+1}), \ldots, s(F_{r(n)}))$ outputs $s(F_j)$ on more than $1/(q(n) * r(n))$ fraction of all z's of size $s(n)$. Therefore $P'(F_1, F_2, \ldots, F_{r(n)}, s(F_1), \ldots, s(F_{j-1}), s(F_{j+1}), \ldots, s(F_{r(n)}))$ outputs $s(F_j)$ with probability $\geq 1/(q(n) * r(n)) \geq 1/p(n)$, which is a contradiction. ∎

Using a combination of techniques used in Lemmas 1 and 3, and Theorems 3, 5 and 6 we can also prove the following theorem :

Theorem 7 *If PH does not collapse, then* for all *$\epsilon > 0$, any probabilistic protocol for $MAXINDSET$, $MAXCLIQUE$ or $MINCOVER$ requires more than $n^{1-\epsilon}$ counterexamples on infinitely many n-vertex graphs.*

6 Conclusion

In this paper we have established a lemma about the structure of satisfiable formulas. From the lemma, we get two very interesting facts about the structure of SAT, provided that the world is as we believe it is (PH is infinite). Firstly, there are large collections A of equal size formulas whose satisfying assignments are independent *i.e.* a satisfying assignment of one provides no information useful to effectively compute that of any other formula in A. Secondly, there are formulas with almost linear number of satisfying assignments which are independent. We have used these to establish a lower bound on the number of counterexamples required to find the lexicographically largest satisfying assignment for boolean formulas. We then used the techniques developed here to prove almost optimal lower bounds on the number of counterexamples required to compute the optimum solution for several NP-optimization problems.

It is still an open question if there are boolean formulas with superpolynomial (in fact, even linear) independent satisfying assignments. More generally are there NP-optimization problems which require a superpolynomial number of counterexamples.

Acknowledgements:

We would like to thank Radha Jagadeesan for correcting the proof of the all important Lemma 1. The authors would also like to thank Juris Hartmanis for his guidance and support.

References

[AMSP80] G. Ausiello, A. Marchetti-Spaccamela, and M. Protasi. Toward a unified approach for the classification of NP-complete problems. *Theoretical Computer Science*, 12:83–96, 1980.

[GJ79] M.R. Garey and D. Johnson. *Computers and Intractability.* Freeman, San Fransisco, 1979.

[KPS90] J. Krajíček, P. Pudlák, and J. Sgall. Interactive Computation of Optimal Solutions. In *Mathematical Foundations of Computer Science*, Springer-Verlag *LNCS* #452, pages 48–60, 1990.

[Kre88] M.W. Krentel. The Complexity of Optimization. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[Pap84] C. Papadimitriou. On the complexity of unique solutions. *Journal of the ACM*, 31:392–400, 1984.

[PY88] C. Papadimitriou and M. Yannakakis. Optimization, Approximation and Complexity Classes. In *20th ACM Symposium on Theory of Computing*, pages 229–234, 1988.

[Sch89] U. Schöning. Probabalistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39(1):84–100, 1989.

[VV86] L.G. Valiant and V.V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986.

[Yap83] C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287–300, 1983.

Appendix

Lemma 5 *If $USAT \in BP \cdot (coNP/poly)$ then PH collapses to Σ_3^{P}.*

The proof of this lemma relies on the following facts:
Fact 1: *coNP/poly* is closed under majority reductions.
Proof: Given any language L we define the languages

$$MAJ(L) = \{< x_1, x_2, \ldots, x_{2n+1} > \ \mid \ \text{a majority of the } x_i's \ \text{ are in } L\}$$

$$MIN(L) = \{< x_1, x_2, \ldots, x_{2n+1} > \ \mid \ \text{a minority of the } x_i's \ \text{ are in } L\}$$

We have to show that for all $L \in coNP/poly$, $MAJ(L) \in coNP/poly$. It is well known that $L \in coNP/poly \Leftrightarrow L \in coNP^S$ for some sparse set S. Now $L \in coNP^S \Rightarrow \overline{L} \in NP^S \Rightarrow MAJ(\overline{L}) \in NP^S \Rightarrow MIN(L) \in NP^S \Rightarrow \overline{MIN(L)} \in coNP^S \Rightarrow MAJ(L) \in coNP^S \Rightarrow MAJ(L) \in coNP/poly$
Fact 2: There is a random reduction T which reduces SAT to $USAT$ with a two sided error of at most $1/2 - 1/16n$.

Proof: In [VV86] it was shown that there is a one-sided random reduction R which reduces SAT to $USAT$ with probability $1/4n$. That is

$$x \in SAT \Rightarrow \mathrm{Prob}(R(x) \in USAT) \geq 1/4|x|$$

$$x \in \overline{SAT} \Rightarrow \mathrm{Prob}(R(x) \in \overline{USAT}) = 1$$

On input x, $|x| = n$, T outputs a trivial member of USAT with probability $1/2 - 1/16|x|$. With remaining $1/2 + 1/16|x|$ probability, T simulates the reduction R on x.
Analysis: If $x \in SAT$ then $\mathrm{Prob}(T(x) \in USAT)$ is at least $(1/2 - 1/16|x|) + (1/2 + 1/16|x|) * (1/4|x|)$ which is at least $(1/2 + 1/16|x|)$. If, on the other hand $x \in \overline{SAT}$ then T outputs a member of $\overline{USAT}$ with probability $1/2 + 1/16|x|$.

Proof: Assume $USAT \in BP \cdot (coNP/poly)$. By Fact 1 and Fact 2, this implies that $SAT \in BP \cdot (coNP/poly)$. Fact 1 also implies that $BP \cdot (coNP/poly) = (coNP/poly)/poly = coNP/poly$ [Sch89]. Therefore we get $SAT \in coNP/poly$ which implies that $NP/poly = coNP/poly$ and PH collapses to Σ_3^{P} [Yap83].