Interactive Computations of Optimal Solutions

Jan Krajíček, Pavel Pudlák and Jiří Sgall

Mathematical Institute, ČSAV

Prague, Czechoslovakia

0. Introduction

Recently we have discovered an interesting connection between optimization problems and logical theories which formalize the principle of induction for certain classes of formulas. This research has suggested a new type of interactive or oracle computations. We shall call this type of computations *counterexample computations*.

Assuming $\mathcal{P}\neq\mathcal{NP}$ there are no polynomial time computable functions which construct optimal solutions for optimization problems connected with $\mathcal{NP}$-complete problems. In the counterexample computations one can get extra information, thus some optimization problems connected with $\mathcal{NP}$-complete problems are computable in polynomial time. In particular the optimization problem CLIQUE is computable in such a way. We conjecture that not all optimization problems are computable in this way, in particular we conjecture it for TSP (TRAVELLING SALESMAN PROBLEM). This conjecture would solve an open problem about the logical theories.

The paper is organized as follows. First we define the concept of optimization problems and discuss other formalizations of this concept. We define the counterexample computations of optimization problems. We mention briefly the connections to some questions about logical theories. Then we prove the main result, which is a hierarchy theorem for counterexample computations. In the last section we define reductions between optimization problems which preserve counterexample computations. We prove that there are complete optimization problems, in particular TSP is complete. We prove that CLIQUE is complete in the class of optimization problems whose value is at most polynomial in the size of the input.

1 Optimization problems

Definition

Let Σ be a finite alphabet *An optimization problem* is a binary relation

$R\subseteq\Sigma^*\times\Sigma^*$, a function $\rho:\Sigma^*\to\mathbb{N}$ and a polynomial p such that R is decidable in polynomial time and $\rho(y)$ is computable in polynomial time in the size of y. (We assume that numbers are coded in binary notation, thus the value $\rho(y)$ can be exponential in the size of y.) We call y *a feasible solution to input* x, if $R(x,y)$ and $|y|\leq p(|x|)$. If y an y' are feasible solutions, we say that y' *is better than* y, if $\rho(y')>\rho(y)$. A feasible solution y is *optimal* if there is no better one.

In order to simplify the exposition we shall make the following additional technical assumptions.

(1) Λ is always a minimal feasible solution, i.e. $R(x,\Lambda)$ and $\rho(\Lambda)=0$ for every x.

(2) We shall assume that p(n) is just the trivial polynomial n.

Example 1. CLIQUE

R(x,y) is the relation "y *is a clique in* x",

$\rho(y)$ is "*the size of the clique* y", (if y is not a clique, then it is 0)

Example 2. TSP

R(x,y) is the relation "x *is a graph with edges labelled by numbers and* y *is a tour in* x",

$\rho(y)$ is "*the length of the tour* y", (if y is not a tour then it is 0).

There are several alternative definitions of optimization problems. An unessential generalization is obtained by letting ρ depend not only on feasible solutions but also on inputs. There is a less trivial generalization in which the function is replaced by a polynomial time decidable quasiordering on the set of feasible solutions to x. A different definition was used by Krentel [K]: *An $\mathcal{NP}$ metric Turing machine* N is a nondeterministic polynomially-time bounded Turing machine such that every branch writes a binary number and accepts. The binary number is the value of the particular feasible solution implicitly computed by the computation branch. Since we need the actual feasible solutions, not only their values, this definition is not suitable for us. Another approach was considered by Papadimitriou and Yannakakis [PY]. They represent optimization problems by formulas $\psi(\mathbf{x},\mathbf{y},G,S)$, where ψ is a first order formula with second order variables G and S. The goal is to maximize

$$\max_S |\{\ \mathbf{x}\ ;\ \exists\mathbf{y}\ \psi(\mathbf{x},\mathbf{y},G,S)\ \}|.$$

2. Counterexample computations

Interactive computations have been introduced by Babai [B] and Goldwasser, Michali, Rackoff [GMR]. The basic idea there is to generalize the concept of nondeterminism. Nondeterministic computation is viewed as a proof. One can use also a dialogue to persuade someone about the truth, this is called *an interactive proof*. Our approach is based on a different idea. In counterexample computations there is also interaction, but perhaps, they are closer to *oracle computations*.

We shall describe a *counterexample computation* as a two player game. One will be called STUDENT, the other one TEACHER. A model situation is an oral examination. STUDENT has limited ability, he can perform only polynomial time computations; TEACHER has unlimited ability, she even knows the strategy of STUDENT. The goal of STUDENT is to compute an optimal solution to a given input x. He can make the conjecture that a feasible solution y (he can always test its feasibility) is optimal and present his conjecture to TEACHER. TEACHER has to tell STUDENT whether y is an optimal solution and if it is not, she must show him a better feasible solution, i.e. *a counterexample* to his conjecture. Then STUDENT can compute another conjecture etc. The computation ends, when STUDENT finds an optimal solution. The objective of TEACHER is to help STUDENT as little as possible (in order to test his knowledge).

We are interested only in computations in which the total computation time of STUDENT is polynomial in the input size.

Example 3

To compute an optimal solution for CLIQUE, STUDENT can use the trivial strategy: the first conjecture is Λ, and then STUDENT just repeats the answers of TEACHER. Clearly such a trivial strategy does not work for TSP, since there can be exponentially many feasible solutions with different values.

We have only the following, a little artificial, example of an optimization problem solvable using a nontrivial strategy of STUDENT. Let G be a 3-regular graph (i.e. each vertex has degree 3). Then, by a well-known theorem, for every edge e of G, there are an even number of Hamiltonian circuits through e. Moreover, if e and a Hamiltonian circuit through e is given, then another Hamiltonian circuit through e can be constructed in

polynomial time.

Example 4

R(x,y) is the relation "y *is a string of at most* 2 *Hamiltonian circuits in a graph* x"

$\rho(y)$ is "*the number of circuits in* y".

By the result mentioned above, STUDENT needs just one query to find an optimal solution.

It is possible (but we cannot prove it) that TEACHER can always force STUDENT to play the trivial strategy. TEACHER knows what STUDENT can compute from her answer y, thus she can pick up a better solution, if STUDENT is able to improve y.

3. Applications to weak arithmetical theories

Here we shall very briefly describe a relation of the above concepts to some research into logical theories formalizing arithmetic. Usually these theories are called fragments of arithmetic, (because by Gödel's theorem there is no recursive theory in which all true arithmetical theorems are provable). There have been defined many fragments of arithmetic. From the point of view of complexity theory the following two theories belong to the most important ones. In order to introduce them we need a class Σ_1^b of formulas which define all $\mathcal{NP}$ sets. We shall not define this class here. Both theories consist of some finite set of simple basic axioms and an induction schema for Σ_1^b formulas. Roughly speaking they formalize induction for $\mathcal{NP}$ sets.

The first one is denoted by S_2^1 and has the schema for Σ_1^b formulas

$$\phi(0) \;\&\; (\forall x)(\phi(\lfloor x/2 \rfloor) \Rightarrow \phi(x)) \Rightarrow (\forall x)\phi(x).$$

The second one is denoted by T_2^1 and has the usual schema of induction for Σ_1^b formulas

$$\phi(0) \;\&\; (\forall x)(\phi(x) \Rightarrow \phi(x+1)) \Rightarrow (\forall x)\phi(x).$$

The theory S_2^1 is strong enough to define polynomial time computable functions and prove many basic properties of them. In a sense the induction schema of S_2^1 is more constructive. Suppose we are able to verify the assumption $\phi(0)$ & $(\forall x)(\phi(\lfloor x/2 \rfloor) \Rightarrow \phi(x))$. Then in order to verify that ϕ holds for some number *a*, we need to apply our verification procedure about log *a* times, which is the size of the representation of *a*. Hence if $\phi(0)$ & $(\forall x)(\phi(\lfloor x/2 \rfloor) \Rightarrow \phi(x))$ can be verified in polynomial time, then $\phi(a)$ can be verified in polynomial time too. In T_2^1 we would have to apply the verification procedure a times, which is exponential in the input size.

It is well known that S^1_2 is contained in T^1_2, but it is an open problem, whether they are equivalent. The following conjecture implies that they are not equivalent.

Conjecture
There exists an optimization problem whose optimal solutions cannot be computed by counterexample computations in polynomial time.

We shall show below that TSP is a complete optimization problem with respect to reductions which preserve counterexample computations, hence the conjecture is equivalent to the statement that TSP *cannot be computed in such a way*.

The counterexample computations that we have introduced are closely related to *Herbrand's theorem* and *no counterexample interpretations* of Kreisel.

4. A hierarchy theorem

In this section we address the question: how many counterexamples STUDENT needs to compute an optimal solution. For each function $f:\mathbb{N}\rightarrow\mathbb{N}$, we consider the class of optimization problems whose optimal solutions can be computed using at most $f(n)$ counterexamples, where n is the size of the input. We denote this class of optimization problems by $\mathscr{CE}(f(n))$. Let us repeat that we consider only counterexample computations which run in polynomial time. (Thus the maximal number of counterexamples is always implicitly bounded by a polynomial.) We should also stress that $\mathscr{CE}(f(n))$ is neither a class of decision problems nor a function class. We cannot define $\mathscr{CE}(f(n))$ as a function class for two reasons: (1) optimal solutions need not be unique, (2) the computation on x depends on the set of feasible solutions to x, a feasible solution for a different input may provide a piece of information which is not available otherwise.

Theorem 1
Let $\varepsilon>0$ be given. Let f be a polynomial time constructable function such that $1\leq f(n)\leq n^{1-\varepsilon}$ for every n. Suppose that each optimization problem which can be computed with $f(n)$ counterexamples can also be computed with only $f(n)-1$ counterexamples, i.e. $\mathscr{CE}(f(n))=\mathscr{CE}(f(n)-1)$. Then $\mathscr{NP}$ problems have polynomial size Boolean circuits

By a result of Karp and Lipton [KL] we get the following corollary

Corollary 1
If $\mathscr{CE}(f(n))=\mathscr{CE}(f(n)-1)$, where f is as above then the Polynomial Hierarchy collapses on the second level, i.e. $\Sigma^p_2=\Pi^p_2$.

Proof of Theorem 1
Let ε and f be given; assume $\mathscr{CE}(f(n))=\mathscr{CE}(f(n)-1)$. We shall prove the following claim.

Claim. For every $R(x,y)$ in $\mathscr{P}$, there exists a function g computable in polynomial time with a polynomial advice such that

$$|y|\leq|x|\,\&\,R(x,y) \Rightarrow R(x,g(x))$$

for every $x,y\in\Sigma^*$.

The computability with a polynomial advice means that there exists a polynomial time computable function $G(x,z)$ and a polynomial $p(n)$ such that

$$(\forall m)(\exists z)(|z|\leq p(m)\ \&\ (\forall x)(|x|=m \Rightarrow g(x)=G(x,z)));$$

(z is a *polynomial advice* for inputs of size m). It is well-known that this is equivalent to the condition that g has polynomial size Boolean circuits. Since R has polynomial size circuits, it follows from the claim that every $\mathscr{NP}$ set of the form $\{\ x\ ;\ (\exists y,|y|\leq|x|)R(x,y)\ \}$ has polynomial size circuits. Now it is an easy exercise to conclude that *every* $\mathscr{NP}$ set has polynomial size circuits. Hence the theorem follows from the claim.

We shall prove the claim. Let R be given. Define an optimization problem S,ρ by:

$$S(u,v) \equiv_{df} v=(v_1,\dots,v_a),\ u=(u_1,\dots,u_b,d),\ a\leq b=f(|u|),$$
$$R(u_1,v_1)\&\qquad \&R(u_a,v_a$$
$$\rho(v) =_{df} a,\ \text{(for which } v=(v_1,\dots,v_a)).$$

Here we assume that we have a suitable coding of sequences so that

$$|u| = |u_1|+\ \dots\ +|u_b|+|d|,$$

(say, using an extended alphabet), and $v=\Lambda$ for $a=0$. The optimization problem S,ρ can be described as follows. An input u consists of $f(|u|)$ "inputs" for R and a padding d. The goal is to find as many elements v_i with $R(u_i,v_i)$ as possible.

Clearly S,ρ is solvable using the trivial strategy with $f(n)$ counterexamples. Hence, by the assumption $\mathscr{CE}(f(n))=\mathscr{CE}(f(n)-1)$, it is solvable with only $f(n)-1$ counterexamples. Let us take such a strategy for

STUDENT. Let m be given. Using this strategy we construct an advice for inputs of size m.

Let $n = m^{\lceil \varepsilon^{-1} \rceil}$. Then n is at most polynomially larger than m and

$$f(n).m \le f(n).n^{\varepsilon} \le n.$$

Let

$$V_1 = \{ x ; |x|=m \ \& \ (\exists y, |y| \le |x|) R(x,y) \}.$$

We choose some function $w(x)$ such that $R(x,w(x))$ holds for every $x \in V_1$. Let $b=f(n)$. We shall define a function α which assigns a number k, $k<b$, to every sequence $u=(u_1, \dots ,u_b)$ with the following property:

(*) There exists a polynomial time computable function β such that if $u=(u_1, \dots ,u_b)$ and $\alpha(u)=k$, then $R(u_{k+1},y)$ holds for $y = \beta(u_1, \dots ,u_b,w(u_1), \dots ,w(u_k))$.

This can be expressed less formally by saying: if $\alpha(u)=k$, then some y with $R(u_{k+1},y)$ can be constructed in polynomial time from $u_1,\dots,u_b,w(u_1),\dots,w(u_k)$.

To define α we shall use STUDENT's strategy. Let $u_1,\dots,u_b \in V_1$ be given. We have

$$|u_1| + \dots + |u_b| = f(n).m \le m.$$

Take $d \in \Sigma^*$ so that $|u_1| + \dots + |u_b| + |d| = n$ and let $u = (u_1,\dots,u_b,d)$. Thus $|u| = n$. We consider the counterexample computation on u in which TEACHER uses a "minimal" strategy: if $v=(v_1,\dots,v_k)$ is STUDENT's last conjecture and $k<b$, then TEACHER will produce $(w(u_1),\dots,w(u_{k+1}))$ as the next counterexample. STUDENT must eventually construct some $v=(v_1,\dots,v_b)$, but since he can ask at most b-1 times, there must be some k, $0 \le k<b$ such that after asking k queries, he constructs a conjecture $(v_1,\dots,v_t)$ with t *larger than* k. Since $(v_1,\dots,v_t)$ must be a feasible solution to u, we have, in particular, $R(u_{k+1},v_{k+1})$. We define $\alpha(u)=k$. In order to obtain the condition (*) we define

$$\beta(u_1, \dots ,u_b,w(u_1), \dots ,w(u_k)) = v_{k+1}.$$

Since STUDENT's strategy is polynomial time computable, so is β; (the value of β for inputs which are not of this form is irrelevant).

Let Q be some b-1 element subset of V_1 and let $x \in V_1$. We say that the pair (Q,x) *is good*, if there is an ordering $u_1,\dots,u_{b-1}$ of the elements of Q such that

$$\alpha(u_1, \dots ,u_k,x,u_{k+1}, \dots ,u_{b-1}) = k.$$

Define a sequence $V_1,V_2 \dots$ of subsets of V_1 and a sequence $Q_1,Q_2 \dots$ of b-1 element subsets of V_1 as follows. Suppose $V_1,\dots,V_j$ and $Q_1,\dots$ has already been defined. We take Q_j to be a b-1 element set such that

$$|\{ x \in V_j ; (Q_j,x) \text{ is good} \}|$$

is maximal and define

$$V_{j+1} := V_j \setminus \{ x \in V_j ; (Q_j,x) \text{ is good} \}.$$

We shall show the following inequality

$$|V_j| \le \left(\frac{b-1}{b}\right)^{j-1}.K^m + \sum_{i=1}^{\infty} \left(\frac{b-1}{b}\right)^i$$

where K is the size of the alphabet Σ. We shall use induction. For j=1 the inequality is trivial. Suppose it holds for some j. First observe that there are at least $\binom{|V_j|}{b}$ good pairs This is because each b-element set $\{u_1,\dots,u_b\} \subseteq V_j$ produces the good pair $(\{u_1,\dots,u_{k-1},u_{k+1},\dots,u_b\},u_k)$ for $k=\alpha(u_1,\dots,u_b)$. (Different orderings of this set may produce more than one good pair.) Dividing this number by the number of b-1 element subsets of V_j we conclude that there is a b-1 element subset Q which forms good pairs with at least

$$\binom{|V_j|}{b} . \binom{|V_j|}{b-1} \quad \frac{|V_j| - b + 1}{b}$$

elements of V_j Thus

$$|V_{j+1}| \quad |V_j| - \frac{|V_j - b + 1}{b} = \frac{b-1}{b}.(|V \quad + 1) \le$$

$$\le \frac{b-1}{b}.\left[\left(\frac{b-1}{b}\right)^{j-1}.K^m + \sum_{i=1}^{\infty}\left(\frac{b-1}{b}\right)^i + 1 \right] = \left(\frac{b-1}{b}\right).K^m + \sum_{i=1}^{\infty}\left(\frac{b-1}{b}\right)$$

which ends the proof of the inequality. Hence for every

$$|V_j| \le \left(\frac{b-1}{b}\right)^{j-1}.K^m + b .$$

Now we define the advice z. Put $t := b.m.\mathit{ln}\ K$. Since $\left(\frac{b-1}{b}\right)^b < \frac{1}{e}$, we have $|V_{t+1}| < 1 + b$. The advice z will be the sequence of pairs $(x,w(x))$, where x runs through all elements of $Q_1 \cup Q_2 \cup \dots \cup Q_t \cup V_{t+1}$. Recall that $b=f(|u|)=f(n)$ and $n = m^{\lceil \varepsilon^{-1} \rceil}$, hence the advice has the size polynomial in m. It remains to check the condition of the claim, i.e. if z is given, then for every $x \in V_1$ we can compute in polynomial time some y, such that $|y| \le m$ and $R(x,y)$. By the construction, every $x \in V_1$ either forms a good pair with some Q_i, $i \le t$, or $x \in V_{t+1}$. If $x \in V_{t+1}$, then we can take $y=w(x)$, and this is easily computable, since $(x,w(x))$ is in z. If (Q_i,x) is a good pair, then

$$\alpha(u_1, \dots ,u_k,x,u_{k+1}, \dots ,u_{b-1}) = k,$$

where $Q_i=\{u_1,\dots,u_{b-1}\}$. Hence we have $R(x,y)$ for

$$y = \beta(u_1, \ldots, u_b, w(u_1), \ldots, w(u_k)).$$

Such a y can be constructed in polynomial time from z, since the pairs $(u_1,w(u_1)), \ldots, (u_k,w(u_k))$ are in z and β is polynomial-time computable. This finishes the proof of the claim, hence of the theorem. □

5. Complete problems

There are several possible definitions of reductions between optimization problems which preserve counterexample computations, we shall choose the most natural one. Different reductions between optimization problems have been considered also in [K] and [PY].

Definition

(i) An optimization problem R,ρ is *reducible* to an optimization problem S,σ, if there exist polynomial-time computable functions $\alpha:\Sigma^* \to \Sigma^*$, $\beta,\gamma:\Sigma^* \to \Sigma^*$ such that for every $x,y,y' \in \Sigma^*$

(1) $R(x,y) \Rightarrow S(\alpha(x),\beta(x,y))$;

(2) $S(\alpha(x),y') \Rightarrow R(x,\gamma(x,y'))$;

(3) $R(x,y)$ & $S(\alpha(x),y')$ & $\sigma(\beta(x,y)) \le \sigma(y'). \Rightarrow \rho(y) \le \rho(\gamma(x,y'))$.

(ii) S,σ is *complete*, if every R,ρ is reducible to S,σ.

Proposition 1

(i) The reducibility is transitive.

(ii) Let an optimization problem R,ρ be reducible to an optimization problem S,σ, and suppose that optimal solutions for S,σ can be computed using counterexample computations. Then optimal solutions for R,ρ can be computed using counterexample computations too.

Proof

Both statements are easy to prove. Let us just sketch the idea of (ii). We have a strategy for STUDENT on S,σ and we have some TEACHER for R,ρ. Use γ to translate STUDENT's conjectures to R,ρ and use β to translate TEACHER's counterexamples to S,σ. □

We define more optimization problems.

MAXSAT

$R(x,y)$ is the relation "x is a propositional formula in CNF and y is an assignment to propositional variables which makes x true";

$\rho(y)$ is the number whose binary expansion gives y.

MAX3SAT

is the same as MAXSAT with x being 3-CNF

Theorem 2

MAXSAT, MAX3SAT and TSP are complete optimization problems

We omit the proof, since it uses standard techniques and it is simila to a proof in [K]. (Note that Krentel uses different concepts.)

Corollary 2

If there is an optimization problem which cannot be computed using counterexample computations, then MAXSAT, MAX3SAT and TSP cannot be computed using counterexample computations.

Definition

An optimization problem R,ρ is called a *polynomial value optimization problem*, if there exists a polynomial p such that, for every $x,y \in \Sigma^*$,

$$R(x,y) \Rightarrow \rho(y) \le p(|x|).$$

Clearly, every polynomial value optimization problem is computable using counterexample computations. CLIQUE is a typical example of such a problem.

Theorem 3

CLIQUE is complete among the polynomial value optimization problems.

Proof

Using a similar argument as in the proof of Cook's theorem, one can show that the following version of MAXSAT is complete among the polynomial value optimization problems:

$R(x,y)$ is the relation "x is a propositional formula in CNF and y is an assignment to propositional variables which makes x true" and $y=(p,q)$, $|p| = \lceil \log_2 |x| \rceil$;

$\rho(y)$ is the number whose binary expansion gives p.

Let us call this problem MAXSAT-LOG. We reduce MAXSAT-LOG to CLIQUE-VAL which is the version of CLIQUE in which vertices have weights. The reduction of CLIQUE-VAL to CLIQUE is obtained easily by blowing up the vertices.

Let x be a formula $\phi(p,q)$ of the form

$$\bigwedge_{i\in I}\ \bigwedge_{j\in J_i} u_{ij}$$

Let $t=|p|=\lceil \log_2|\phi|\rceil$. We define a graph G and a valuation on vertices of G. The vertices of G are numbers $1,\ldots,t$, and pairs (i,j) for $i\in I$, $j\in J_i$. The edges are

(i) all (h,k) for $1\le h,k\le t$;

(ii) $((i,j),(a,b))$ iff $i\neq a\ \&\ (\forall h\le t)(\ \{u_{ij},u_{ab}\}\neq\{p_h,\neg p_h\}\ )$;

(iii) $((i,j),h)$ iff $u_{ij}\neq\neg p_h$.

The weight of each vertex (i,j) is 2^{t+1}, the weight of each vertex $h\in\{1,\ldots,t\}$ is 2^h. The verification of the properties of the reduction is left to the reader. □

Corollary 3

Suppose that Polynomial Hierarchy does not collapse. Then for each of the problems MAXSAT, MAX3SAT, TSP and CLIQUE, there exists $\varepsilon>0$ such that every counterexample computation requires at least $n^{1-\varepsilon}$ counterexamples for infinitely many inputs, where n is the input size.

Proof

Suppose that Polynomial Hierarchy does not collapse. Then, by Theorem 1, there exists an optimization problem which is computable using, say, $\lceil n^{1/2}\rceil$ counterexamples, but which cannot be computed using less than $\lceil n^{1/2}\rceil$ counterexamples. A closer look at the proof of Theorem 1 reveals that actually there must be an infinite number of inputs which require $\lceil n^{1/2}\rceil$ counterexamples. Since the reductions preserve the number of counterexamples and can increase the size of the input at most polynomially, we get the statement of the corollary from Theorem 2. For CLIQUE we only have to check that the optimization problem from Theorem 1 has polynomial values and apply Theorem 3. □

Assuming our conjecture that there are optimization problems whose optimal solutions are not computable using counterexamples, Corollary 3 is interesting only for CLIQUE. However there is still a big gap between the lower and the upper bounds for this problem.

Proposition 2

Optimal solutions for CLIQUE can be computed using $n-\sqrt{n}+1$ counterexamples

Proof

Let G be a given graph. STUDENT will compute as follows. He constructs a maximal clique (i.e. a clique which cannot be extended to a larger one), deletes the vertices of the clique from G, constructs a maximal clique in the remaining part etc. until all vertices are exhausted or an empty graph (i.e. an independent set in G) remains. STUDENT's first conjecture will be a clique of maximal size from these cliques. In the next rounds he just uses the trivial strategy (repeats TEACHER's answers). We shall show that in $n-\sqrt{n}+1$ rounds a maximal size clique is constructed. Consider the following cases.

(1) The remaining independent set has size $>\sqrt{n}$. Then the maximal size clique has size $<n-\sqrt{n}$, hence there can be at most $n-\sqrt{n}-1$ interactions.

(2) If one of the maximal cliques constructed at the beginning has size $\ge\sqrt{n}$, then again there can be at most $n-\sqrt{n}$ interactions.

(3) Suppose each of the constructed maximal cliques has size $<\sqrt{n}$ and the remaining independent set has size $\le\sqrt{n}$. Then the number of these maximal cliques must be at least $\sqrt{n}$. Note that any maximal clique of G, in particular any maximal size clique in G, can contain at most one of the constructed maximal cliques as a subset. Hence the maximal size of a clique in G is at most $n-\sqrt{n}+1$ and we can conclude as in the former cases. □

6. Conclusions

Our results indicate that optimization problems like MAXSAT and TSP are more difficult than polynomial value optimization problems, like CLIQUE. This supports our conjecture that there are optimization problems whose optimal solutions cannot be computed using counterexample computations. In a different setting Krentel shows that TSP is really more difficult than CLIQUE (assuming, of course, $\mathcal{P}\neq\mathcal{NP}$). He defines a function class $\mathcal{FP}^{SAT}[z(n)]$ as follows. A function f is in $\mathcal{FP}^{SAT}[z(n)]$, if f(x) can be computed in polynomial time using at most $z(|x|)$ times an oracle for the $\mathcal{NP}$-complete problem SAT. He proves that the value of the maximal size clique can be computed in $\mathcal{FP}^{SAT}[O(\log n)]$, while there are optimization problems for which we need more. The point is that if we are interested only in computing *the values* of optimization problems, we need only O(log n) bits to determine the value of an instance of a polynomial value optimization problem. In our setting we want to compute *optimal solutions*. Then we need more than O(log n) bits to determine them even for CLIQUE. Hence we cannot use Krentel's results.

We believe that our hierarchy theorem should hold also for functions f(n) which are closer to n or even larger than n. For f(n) growing faster than every polynomial, we do not get new classes, since the number of counterexamples is always implicitly bounded by a polynomial. Perhaps a better definition of $\mathscr{CE}$(f(n)) might be the following. Instead of requiring that the total running time of STUDENT is polynomial, we should require that there is a fixed polynomial such that the running time before the first conjecture and between each two consecutive conjectures is bounded by this polynomial. Then it might hold that TSP required exponentially many counterexamples.

References

[B] L.Babai, *Trading group theory for randomness*, 17-th STOC, 1985, pp. 421-429.

[GMR] S.Goldwasser, S.Michali, C.Rackoff, *The knowledge complexity of interactive proof systems*, 17-STOC, 1985, pp.291-304.

[KL] R.M.Karp, R.J.Lipton, *Some connections between nonuniform and uniform complexity classes*, 12-th STOC, 1980, pp.302-309.

[KPT] J.Krajíček, P.Pudlák, G.Takeuti, *Bounded Arithmetic and Polynomial Hierarchy*, Annals of Pure an Applied Logic, to appear.

[K] M.Krentel, *The complexity of optimization problems*, 18-th STOC, 1986, pp.69-75.

[PY] C.H.Papadimitriou, M.Yannakakis, *Optimization, approximation, and complexity classes*, 20-th STOC, 1988 pp.229-234.