Issues in NP-Optimization and Approximation

Desh Ranjan
Ph.D Thesis

92-1297
August 1992

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

ISSUES IN NP-OPTIMIZATION AND APPROXIMATION

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Desh Ranjan
August 1992

ISSUES IN NP-OPTIMIZATION AND APPROXIMATION

Desh Ranjan, Ph.D.

Cornell University 1992

Optimization or finding the best solution for a problem amongst several possible ones is one of the central themes in computing. In particular, NP-optimization (NPO) problems, examples of which include such well-known problems like Integer Programming and Traveling Salesperson Problem, have proved to be of great practical and theoretical importance. Different NPO problems exhibit starkly different properties and understanding the structure of these problems and their classification has been a long-standing goal in theoretical computer science.

This thesis investigates the properties of NPO problems in two settings. In the first part of the thesis we investigate how the logical expressibility of NPO problems relates to some of their computational properties like approximability and self-improvement. In the second part we study NPO problems in the context of a relatively new model called the *counterexample model*. This allows us to achieve two objectives : Firstly, it gives us a framework to study and analyze incremental computation of optimal or near-optimal solutions in an abstract setting. This is useful because, in practice, for most of the NPO problems, one has to resort to inexact algorithms which work incrementally towards computing a good solution.

Secondly, it gives us a way to precisely formulate and study questions about the structure of these problems which we believe are fundamental from theoretical point of view - for example, how much does the knowledge of one solution of a problem help in computing another solution?

Biographical Sketch

Desh Ranjan was born on June 15, 1965 in Lucknow which lies in the fertile Gangetic plains of India. After a religious and relatively uneventful childhood and adolescence, which included schooling at Mahanagar Boys' High School and Colvin Taluqedars' College in Lucknow, Desh joined the Indian Institute of Technology at Kanpur from where he recieved his Bachelor's degree in Computer Science in May 1987. Thereafter, in search of truth, beauty, broader perspective and a higher degree, Desh joined the graduate school at Cornell University where he succeeded in achieving three out of these four goals. Desh was awarded a Master's degree in computer science in August 1990 and a Ph.D. in August 1992. Despite the long, cold Ithaca winters which prohibit outdoor activity and can make one believe in divine retribution, Desh is now an atheist and a decent tennis player.

To my parents

Acknowledgements

I am greatly indebted to Juris Hartmanis for guiding me through this thesis. This thesis wouldn't have been completed without his constant encouragement. I thank Devika Subramanian and Richard Shore for serving on my special committee and providing many helpful suggestions.

It's my pleasure to acknowledge the help of my coauthors Suresh Chari, Alessandro Panconesi and Pankaj Rohatgi. Parts of this thesis are as much their work as mine. Special thanks to Alessandro for introducing me to the subject of NP-optimization and approximation, which is what this thesis is all about. I have also benefited from discussions with Tushar Chandra, Richard Chang, Radhakrishnan Jagadeesan, Jim Kadin, Dexter Kozen, Steve Mitchell, Prakash Panangaden, Steve Vavasis and Vijay Vazirani.

I would like to thank all my friends (I like to believe that there are too many of them to name) for reasons as diverse as they are. I would specially like to thank Daniela Rus and Judith Underwood, my officemates for the last year at Cornell, for patiently putting up with all the complaining I had to do about jobs or lack thereof.

I gratefully acknowledge the financial support provided by Cornell University (Sage Graduate Fellowship), Mathematical Sciences Institute (MSI Fellowship) and National Science Foundation (Research Grants #DCR-8520597 and #CCR-8823053) to facilitate my research.

Last but not the least, I thank Jan Batzer, Becky Personius, Cindy Robinson-Hubbell and Suzy Harris for taking care of multitude of odd jobs that could have gobbled up significant amount of my time otherwise, and buoying up my spirits with their mild bantering and cheerful disposition on innumerable occasions.

Table of Contents

List of Figures

Chapter 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 or approximate solutions of what are called NP-optimization (hereafter NPO) problems is of great interest and importance in practice and theory. Although the historical importance of the study of NP-optimization in the development of the field of computer science hardly requires recounting, we shall do so briefly.

After the first computers were built, substantial effort was spent on developing fast algorithms for solving various optimization problems. Typical of these were circuit design for computing boolean functions [Sha49,Yab59a] and various linear programming and graph theoretic problems, a classic example of such a problem being the traveling salesperson problem, MINTSP, which is the problem of computing the shortest hamiltonian tour in a given edge-weighted graph. It was observed that even though these problems were effectively solvable in theory trivially, since one had to explore only a finite *solution space*, in practice they became intractable very quickly because the size of this solution space grew at a very fast rate as compared to the problem instance size. Hence, even before the notion of time complexity was defined the possibility of "combinatorial explosion" and extreme inefficiency and impracticality of the "perebor" (the brute force search) to solve these problems were discussed [Yab59b]. Combinatorial problems were crucial in shifting the attention from effective computability to feasible computability. It was the combinatorial problems that really brought forth the difference between efficient and inefficient algorithms [Edm65,vN53] and led to formal definition of time complexity. For a nice exposition of how combinatorial optimization problems have played an important part in development of the theoretical computer science, we refer the reader to Karp's Turing award lecture [Kar86].

After the notion of time complexity was formalized by Hartmanis and Stearns [HS65], it was noticed that the best known algorithms for solving many practical and useful combinatorial optimization problems had exponential time complexity. In spite of the failure of all attempts it was initially thought that it was just a matter of time before, with cleverness and subtlety, efficient algorithms were discovered for these problems. The picture changed drastically after the work of Cook and Karp in the early 1970's. In a seminal paper written in 1971, Cook formally defined the classes of languages P and NP, showed that the satisfiability problem is complete for NP and stated the open problem $P \stackrel{?}{=} NP$. This problem that has been central to the development of theoretical computer science over the last two decades[Coo71]. Immediately afterwards, Karp in another influential piece of work showed that the decision versions of many well-known optimization problems were NP-complete[Kar72]. This meant that there were efficient algorithms for solving these problems if and only if P=NP. The statement P=NP has deep implications. For a quick survey of the history and status of the $P \stackrel{?}{=} NP$ question see [Sip92].

These developments had two effects. Firstly, significant research effort in the coming years was devoted to understanding the structure of the class NP. This lead to a good understanding and an elegant theory of NP-completeness capped by the postulation of the *Berman-Hartmanis conjecture* which states that all NP-complete sets are *p-isomorphic.* In other words, all NP-complete sets had more or less the same structure. Secondly, it helped to redirect the research in the area of optimization. In particular, it became quite clear that it would be more fruitful to concentrate efforts on developing fast approximation algorithms rather than exact algorithms. Early on, this resulted in a wealth of results providing ingenious algorithms for approximation of individual problems, and several isolated proofs of non-approximability of others (assuming $P \neq NP$). Very soon it was noted that, contrary to the decision problems, different optimization problems exhibited radically different behavior with respect to many computational properties of interest, *e.g.* approximation, although the reasons why this was the case were far from clear. This situation was correctly deemed unsatisfactory by many computer scientists who directed their efforts towards the classifying NPO problems and providing a unifying theoretical framework to study, classify and understand the diverse behavior of the NPO problems. We shall summarize some of these efforts very briefly in the next chapter.

In this thesis we present our contribution to the general endeavor of understanding the structure of NPO problems. The diversity amongst the various NPO problems arises because of the cost functions associated with the problems. Notice that all the NP decision problems can be thought of as optimization problems with the associated cost function being the function that associates a cost of zero with the non-solutions and a cost of one with the feasible solutions. This, intuitively and mathematically, is a big homogenizing factor and it makes the resulting problems isomorphic. On the other hand, with different costs attached to different solutions of the same problem instance, the NPO problems display widely varying behavior with respect to computationally interesting properties. Hence, understanding the NPO problems basically entails understanding the the structure of the solution space resulting from the imposition of different cost functions on the solution space. This is our goal in this thesis.

The thesis investigates the properties of NPO problems in two different settings. In chapter 3 we investigate the logical expressibility of the NPO problems and the relationship between logical expressibility and some computational properties of interest. The motivation here is to see if important computational properties like approximation can be linked to formal syntactic representations of these problems. In chapter 4 and 5 we study the NPO problems in the context of a new model called the *counterexample model.* Our motivation is to understand the interdependence of the solutions modulo polynomial time computations, and to characterize if and how the knowledge of non-optimal solutions can be used towards computing an optimal or a near-optimal solution efficiently. Chapter 2 covers the background material required for the other chapters.

Chapter 2

Preliminaries

In this chapter we shall give the definitions and background required for the rest of the thesis. We assume knowledge of basic computational complexity theory, graph theory and predicate calculus . Since NP-optimization problems are the primary objects of interest here we shall start by giving a precise formal definition of what an NP-optimization problem is. We shall then illustrate it via a few examples.

2.1 NP-optimization : Definition and Examples

Definition 1 *An* NPO *problem is a tuple* $F = (\mathcal{I}_F, S_F, f_F, opt)$ *where*

- $\mathcal{I}_F \subseteq \Sigma^*$ *is the set of* input instances. *It is recognizable in polynomial time.*
- $S_F(x)$ *is the set of* feasible solutions *on input* $x \in \mathcal{I}_F$. *We require that* $\forall x \in \mathcal{I}_F$, $S_F(x) = \{y \mid |y| \leq q_F(|x|) \wedge \pi_F(x,y)\}$ *where* q_F *is a polynomial and* π_F *is a polynomial time computable predicate.* q_F *and* π_F *depend only on* F.
- $f_F : I_F \times \Sigma^* \to N$, *the* objective function, *is a polynomial time computable function.* $f_F(x,y)$ *is defined only when* $y \in S_F(x)$.
- $opt \in \{\max, \min\}$.

Solving an optimization problem F given the input $x \in \mathcal{I}_F$, means finding a $y \in S_F(x)$ such that $f_F(x, y)$ is optimum. The optimum value of F on input x is defined as

$$opt_F(x) = opt_{y \in S_F(x)} f_F(x, y).$$

We present some examples of how some well-known problems can be expressed in this formalism.

The first example is the MAXCLIQUE problem which is the problem of finding a clique in a given graph which has the maximum number of vertices.

Example 1. MAXCLIQUE

INSTANCE. An undirected graph $G = (V, E)$.

SOLUTIONS. $C \subseteq V$ such that C is a clique in G, *i.e.*, it induces a complete subgraph in G.

COST FUNCTION. $cost(G, C) = \| C \|$.

OBJECTIVE. MAX .

We are assuming that encoding of the graphs is such that it can be recognized whether a given string is an encoding of a graph in polynomial time. This is true for any reasonable encoding of the graphs. Also note that given a graph G and a subset C of its vertices it is possible to check if C is a clique in G in polynomial time. The cost function, which is just the number of vertices in C, is computable in polynomial time. Hence, this specification satisfies all the criteria of the definition.

The next example is the famous traveling salesperson problem. The goal here is to compute a hamiltonian tour of shortest length given an edge-weighted graph with non-negative weights.

Example 2. MINTSP

INSTANCE. A graph $G = (V, E)$ with non-negative weights assigned to edges.

SOLUTIONS. $H \subseteq E$, such that edges in H form a hamiltonian tour in G.

COST FUNCTION. $cost(G, H)$ = sum of the weights of the edges in H.

OBJECTIVE. MIN .

The following two examples revolve around the well-known satisfiability problem which is known to be NP-complete. The first problem is to find an assignment to the variables of a given boolean formula in conjunctive normal form that satisfies the maximum number of clauses of the formula.

Example 3. MAX 3SAT

INSTANCE. A boolean formula F in 3CNF

SOLUTIONS. Assignments A to the variables of F.

COST FUNCTION. $cost(F, A)$ = number of clauses in F satisfied by A.

OBJECTIVE. MAX .

The next example, which we shall be using repeatedly later, is one of the hardest problem amongst the NP-optimization problems. This is the problem of finding the lexicographically largest *satisfying* assignment for a given boolean formula F.

Example 4. LEXMAXSAT

INSTANCE. A boolean formula F on variables $x_1, ..x_n$

SOLUTIONS. Assignments to the variables of F : $a_1, ..a_n$, $a_i \in \{0, 1\}$ such that $F(a_1, ..a_n)$ is true.

COST FUNCTION. $cost(F, A = a_1 \dots a_n)$ = value $a_1 a_2 \dots a_n$ treated as a binary number,*i.e.*, $\Sigma_{i=1}^{n} 2^{n-i} a_i$.

OBJECTIVE. MAX .

It seems possible to define NPO more concisely in the following way. We use $N(x, y)$ to indicate the final output of a nondeterministic Turing machine $N(x)$ along the computation path y. If we interpret $N(x, y)$ as a natural number then $F \in$ NPO iff there exists a polynomial time NDTM N such that, for all $x \in \mathcal{I}_F$,

$$opt_F(x) = \max_y N(x, y).$$

However, this definition is unsatisfactory for our purpose as it does not explicitly state what the set of feasible solutions and the objective function are. We shall be studying approximation in later chapters and we believe that it is essential to separate these two objects if one wants to study approximation.

As stated previously, solving most of the NPO problems exactly in polynomial time is as hard as proving P=NP. So, in practice, one has to compromise and adopt less ambitious approaches. One natural way to do so is to look for algorithms that compute solutions that are approximately optimal. We discuss approximations in chapter 3 and 5.

2.2 Approximation Algorithms and Classes

To formally study the behavior of a problem with respect to approximation, we need to define precisely what it means for an algorithm to compute an approximately optimal solution. For this, we need the notion of relative error for a feasible solution.

Definition 2 *The* relative error *of a feasible solution y of instance x of an* NPO *problem F is defined as*

$$\mathcal{E}_F(x, y) = \frac{|\,opt_F(x) - f_F(x, y)\,|}{opt_F(x)}$$

where $y \in S_F(x)$.

Now we are ready to define the notion of approximability.

Definition 3 *An* NPO *problem F is* ϵ-approximable, *if there exists a polynomial time algorithm A such that, for all instances x of F:* i) $A(x) \in S_F(x)$, *and* ii) $\mathcal{E}_F(x, A(x)) \leq \epsilon$. *A problem is* approximable *if there is an $\epsilon \in (0,1)$ such that it is ϵ-approximable.*

Notice that the above definition is only meant for maximization problems. For minimization problems the definition would be the same except that ϵ could be any real greater than zero.

Definition 4 APX *is the class of all approximable* NPO *problems.*

Examples of problems in APX are MAX SAT, MAX CUT, MIN ΔTSP, MIN BIN PACKING and MIN NODE COVER [GJ79,PS82,Joh74].

NP-optimization problems display diverse behavior with respect to approximation. For example, it is well known that some NPO problems are *ϵ-approximable* for *any* $\epsilon > 0$ [GJ79,PS82].

Definition 5 *An* NPO *problem is said to have a* polynomial time approximation scheme *if there exists an algorithm $A(x, \epsilon)$ such that, for all ϵ and all $x \in \mathcal{I}_F$:* i) $A(x, \epsilon) \in S_F(x)$, *and* ii) $\mathcal{E}_F(x, A(x, \epsilon)) \leq \epsilon$. *The complexity of A must be polynomial for any* fixed ϵ.

To clarify the definition, the complexity of a polynomial time approximation scheme can be something like $2^{1/\epsilon}p(|x|)$ or something like $|x|^{1/\epsilon}$; these cases actually arise in practice [HS87,GJ79].

Sometimes the dependence on ϵ is also polynomial.

Definition 6 *An* NPO *problem is said to have a* fully polynomial time approximation scheme *if it has polynomial time approximation scheme whose complexity is of the kind $p(1/\epsilon, |x|)$, where p is a polynomial.*

Definition 7 PTAS *is the class of* NPO *problems that have a polynomial time approximation scheme.*

Definition 8 FPTAS *is the class of* NPO *problems that have a fully polynomial time approximation scheme.*

Many scheduling problems are known to be in PTAS [HS87]. MAX KNAPSACK is an example of a problem belonging to the class FPTAS [PS82].

The classification of problems above is based on degree of approximability of the problems. NPO problems have been classified also on the basis of other difficulty measures. For example one criteria that has been used for classification is the number of oracle queries a polynomial time machine requires to ask a SAT oracle to compute an optimum solution[Kre88]. Several interesting result have been proved in this area and one of the most interesting open problems in structural complexity, $P^{SAT} \stackrel{?}{=} P^{SAT[log]}$ derives from this work. In the next chapter, we shall define classes of optimization based on logical expressibility and in chapter 4 and 5 we shall define classes of NPO problems based on the difficulty measure used in the counterexample model.

2.3 Reductions: Definitions and Properties

To understand and classify various optimization problems according to amenability to approximation it is convenient to define reductions analogous to the more conventional reductions between languages in complexity theory. Recall that in complexity theory the general idea behind a reduction is that if a language A reduces to B and B is in a particular complexity class $\mathcal{C}$ then A is also in $\mathcal{C}$. Moreover the reduction relation is transitive that is if A reduces to B and B reduces to C then A reduces to C. Intuitively, A reduces to B in some way captures that A is easier than B and that's why this is usually denoted as $A \leq B$.

We would like analogous reductions for NPO problems with the added constraint that these reductions preserve approximability. We define below precisely what this means, but clearly the general idea is that if an NPO problem A reduces to another NPO problem B and B is approximable then so should be A. We note

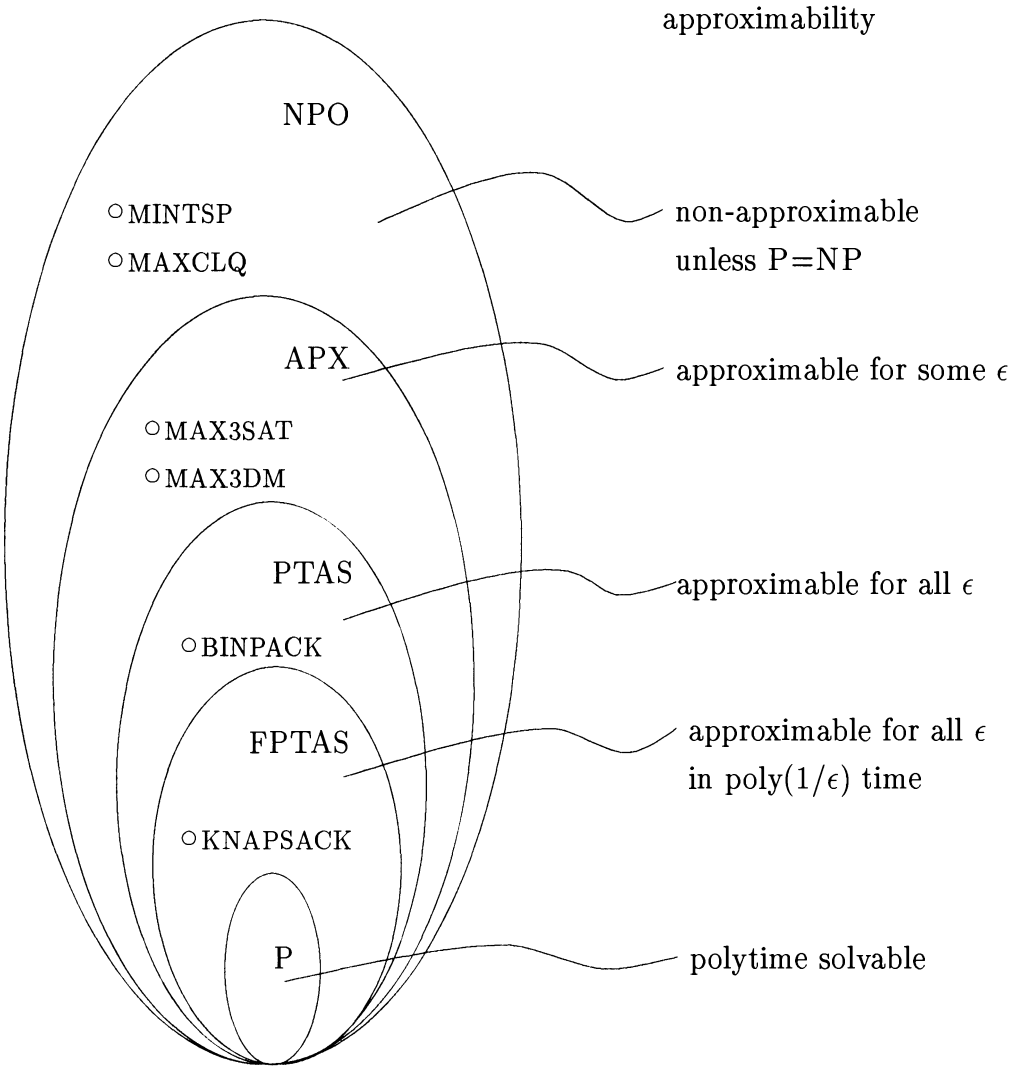

Figure 2.1: Classification of NPO problems based on approximability

that most known reductions to NP-complete problems do not satisfy this property when simply translated to NPO problems.

Definition 9 *Given two* NPO *problems F and G, a* PTAS preserving reduction *(P-reduction) from F to G is a triple $f = (t_1, t_2, c)$ such that*

i) t_1, t_2 *are polynomial time computable functions and* $c : (0,1) \to (0,1)$.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$, *and* $t_2 : \mathcal{I}_F \times S_G(t_1(x)) \to S_F(x)$.

iii) $\forall x \in \mathcal{I}_F$ *and* $\forall y \in S_G(t_1(x))$, *if* $\mathcal{E}_G(t_1(x), y) \leq c(\epsilon)$ *then* $\mathcal{E}_F(x, t_2(x, y)) \leq \epsilon$.

Most of the reductions we use will actually be much stronger than this reduction.

Definition 10 *A P-reduction from F to G is said to be an* approximation preserving reduction *(A-reduction) if* $c(\epsilon) = \epsilon$.

In a P-reduction, we use t_1 to map instances of F into instances of G, and t_2 to map approximate solutions for G back into approximate solutions of F. The relation among t_1, t_2 and c ensures that the following proposition holds.

Proposition 1 *If* $G \in$ PTAS *and* $F \leq_P G$, *then* $F \in$ PTAS.

Proposition 2 *P-reductions compose, i.e. if* $F \leq_P G$ *and* $G \leq_P H$ *then* $F \leq_P H$.

Chapter 3

Logical Expressibility and Approximation[1]

In this chapter we will investigate the logical expressibility of NP-optimization problems and then the relationship between expressibility and approximation. Characterization of complexity classes via logical expressibility is a well-developed area in computational complexity [Fag74,Imm80]. It has yielded a fresh perspective on complexity classes and has lead to breakthrough results like the famous Immerman-Szelepcsényi theorem[Imm88,Sze87].

3.1 Why Logical Expressibility

In order to develop a theory for approximation of NPO problems, one has to define subclasses of NPO with problems in the same subclass having similar approximation properties. Defining these classes in terms of Turing machines presents a fundamental problem; changing something "computationally insignificant" like the value of a single bit, can have enormous effect on the approximation properties of the computed function. It is possible to define approximation classes within NPO in terms of Turing machines as is done by Crescenzi and Panconesi[CP89].

[1] joint work with Alessandro Panconesi

The results are interesting, but it seems doubtful that meaningful problems can be proven complete in those classes.

To avoid the problems which arise in the Turing machine model, Papadimitriou and Yannakakis [PY88] introduced an approach based on the logical characterization of NP given by Fagin [Fag74]; this result states that NP is the set of languages that are the generalized spectra of a second order existential formula, ranging over finite structures. They use this characterization to define a natural class of NPO problems, which they call MAX NP. Roughly speaking, a problem in MAX NP has the property that the set of its feasible solutions can be described by a formula of the type $\exists \bar{y}\ \Phi(\bar{y}, S)$, where Φ is quantifier-free, S is a feasible solution and $\bar{y}$ ranges over the input structure, such as a graph or a boolean formula. Interestingly enough, they show that all the problems in MAX NP are approximable and that there is a *uniform* way in which they can be approximated. In this chapter, our aim is to further investigate the relationship between logical expressibility and computational properties.

3.2 Background

We assume familiarity with basic predicate calculus i.e. inductive definition of formulae, interpretations, models etc. A *finite structure* is a nonempty finite set, along with certain given functions and relations on the set. Finite structures can be used to specify problem instances. For example, the set of finite graphs can be specified by a finite structure (V, E) where V is a finite set (the set of vertices) and E a binary relation (the edge relation). Different graphs can be specified by different assignments to V and E. Similarly, a boolean formula in conjunctive normal form can be described by a finite structure (A, P, N) where A is a finite set of integers and P and N are binary predicates. To describe a formula F with m clauses and n variables we let $A = \{1, 2 \ldots m + n\}$ and assign $P(i, j)$ true iff variable i occurs positively in clause j and assign $N(i, j)$ true iff variable i occurs negatively in clause j. For a more formal treatment of finite structures we refer to [Fag74].

An important result in computational complexity is the logical characterization of NP due to Fagin [Fag74]; a language is in NP if and only if it is the *generalized spectrum* of a second order existential formula, ranging over finite structures. The generalized spectrum of a formula is the set of finite structures which model the formula *i.e.* that make the formula true.

For example, $\varphi \in$ SAT if and only if

$$\exists T\ \forall c\ \exists x\ (P(x,c) \wedge T(x)) \vee (N(x,c) \wedge \neg T(x))$$

Intuitively, T is a second-order variable that ranges over truth assignments; φ is described by means of the two binary predicates P and N as explained above *i.e.* $P(x,c) = \text{TRUE}$ iff variable x appears positive in clause c and $N(x,c) = \text{TRUE}$ iff variable x appears negated in clause c. The formula $(P(x,c) \wedge T(x)) \vee (N(x,c) \wedge \neg T(x))$ ensures that T sets to true at least one literal for each clause. Hence, SAT is the generalized spectrum of the above formula.

In general, for any language L in NP there is a *quantifier free* formula Φ_L such that

$$I \in L \ \Leftrightarrow\ \exists S\ \forall \overline{x}\ \exists \overline{y}\ \Phi_L(I, S, \overline{x}, \overline{y})$$

(see [Fag74]). Informally, the instance I is described with a finite structure $I = \{A, P_1^{a_1}, .., P_k^{a_k}\}$, where $P_i^{a_i} \subseteq A^{a_i}$, and A is a finite set. In the formula Φ_L, I stands for the set of predicates $P_i^{a_i}$ (this is an abuse of notation; for more formal description see [Fag74]). $S \subseteq A^s$ is a predicate of arity s describing the solution (e.g a satisfying assignment), and $\overline{x}, \overline{y}$ are vectors of fixed arity of elements of A. We could consider a more general format where S too is a collection of predicates; here we consider the case where S is a single predicate for sake of simplicity, but most of our proofs can be generalized.

It is important to realize that the formula Φ_L is the same for all instances I. In particular it is of fixed size, and the arities of the vectors $\overline{x}, \overline{y}$, together with the arities of the predicates appearing in I and S are fixed.

Papadimitriou and Yannakakis noticed that a similar formalism can be used to describe NPO problems. Again, for sake of clarity, we consider an example. Take the problem MAX SAT: given a boolean formula φ in *CNF*, find an assignment that maximizes the number of clauses set to true.

Let $\Phi(x, c, P, N, T)$ be an abbreviation for $(P(x,c) \wedge T(x)) \vee (N(x,c) \wedge \neg T(x))$.

Then, for all instances φ the following holds

$$opt_{\text{MAX SAT}}(\varphi) = \max_T \| \{c \mid \exists x \; \Phi(x, c, P, N, T)\} \|.$$

Definition 11 *[PY88]* MAX NP *is the class of* NPO *problems* F *such that*

$$opt_F(I) = \max_S \| \{\overline{x} \mid \exists \overline{y} \; \Phi_F(I, S, \overline{x}, \overline{y})\} \|$$

where Φ_F *is a quantifier free formula.*

Theorem 1 *[PY88] Every problem in* MAX NP *is ϵ-approximable for some $\epsilon \in (0,1)$, i.e.* MAX NP $\subseteq$ APX.

This result is important and surprising because it links logical expressibility which is a syntactical notion to approximability which is a computational property and there is no apparent reason why the two should be related. This theorem immediately motivates one to consider the following two questions. The first question is if the reverse direction of the theorem is true or equivalently, if there are approximable problems that are not in this class, that is, what is the expressive power of the class. The second, more general question is: what is the relationship, if any, between the logical representation of a problem and its approximation properties? We shall address these two questions in the following sections.

3.3 Results and Organization

We begin by investigating the expressive power of MAX NP and showing that it is rather limited. We prove that well-known and important problems like MAXCLIQUE, MAX 3DM, and MAX 3SP (optimization versions of 3DM and SET PACKING) are not in MAX NP. It is not known if MAXCLIQUE is approximable[2] but we prove that MAX 3DM and MAX 3SP are approximable. In fact, we also prove the stronger fact that MAX NP does not even capture all of P because we show that the problem of finding a maximum matching in a graph can not be expressed in MAX NP. These are expressibility results and do not rely on any assumptions (such as P $\neq$ NP).

It turns out that all these problems which cannot be expressed as problems in MAX NP have similar logical structure and they fit nicely into a new class that we call MAX Π_1. Loosely speaking, these problems have the property that the set of feasible solutions can be described by means of a first order formula of the type $\forall \bar{y}\ \Phi(\bar{y}, S)$.

We investigate the structure of MAX Π_1 and find natural complete problems under reductions that preserve approximability [CP89,OM87,PM81,PY88]. For example we prove that, given a boolean formula, the problem of finding a satisfying assignment that sets to true the maximum number of variables (we call this problem MAX ONES) is MAX Π_1-complete. MAX Π_1 in its full-fledged form turns out to be too expressive; the complete problems for MAX Π_1 are not approximable unless P = NP. But many natural problems in MAX Π_1 for example MAX 3DM, MAXCLIQUE, MAX 3SP do not seem to be as hard as the complete problems for MAX Π_1. Interestingly enough, neither do they seem to require the full expressive power of MAX Π_1 to prove that they are indeed in MAX Π_1. This leads us to define subclasses of MAX Π_1 based on restricting the structure of the logical formulae allowed to express the problems. The motivation for the constraints imposed

[2]By a recent result of Arora and Safra these problems are not approximable unless P= NP

comes from observing the similarity in the expressions for the problems mentioned above. The major restriction that is imposed corresponds to saying that if S is a feasible solution for the problem and $S' \subset S$, then so is S'. The smallest and most interesting of these subclasses contains MAX k-DM, MAX k-SP, and has MAXCLIQUE and MAX GRAPH k-COLORING as complete problems. The other classes have a natural generalization of MAXCLIQUE as their complete problems. All of the complete problems share the interesting property that either they are non-approximable or are approximable within any fixed ratio.

This chapter is organized as follows. In Section 3.4 we prove that MAXCLIQUE, MAX 3DM, and MAX 3SP, and the maximum matching problem are not in MAX NP. We then introduce the class MAX Π_1 and prove that the above problems belong to it. In Section 3.5 we prove the MAX Π_1-completeness of the problems MAX ONES and MAX NSF with respect to approximation preserving reductions (MAX NSF is the following problem: given a set of *CNF* -formulae, find the maximum number of satisfiable ones). In Section 3.6, we define a subclass of MAX Π_1, the class RMAX, and prove completeness results for several optimization problems.

3.4 Expressiveness of MAX NP

In this section we show that certain important optimization problems are not in MAX NP. In fact, we show that there are approximable problems and polynomially computable problems that are not in MAX NP. We first show that MAXCLIQUE is not in MAX NP; it is not known whether MAXCLIQUE is approximable. Then we introduce two large classes of matching and set packing problems. For these, we prove that they are approximable and that they do not belong to MAX NP. All these problems naturally belong to a new complexity class that we call MAX Π_1.

The following theorem is motivated by a more general principle

$$\mathcal{A} \models \exists x\, \Phi(x) \ \wedge\ \mathcal{A} \subseteq \mathcal{B} \ \Rightarrow\ \mathcal{B} \models \exists x\, \Phi(x)$$

where $\Phi(x)$ is quantifier-free, and $\mathcal{A} \subseteq \mathcal{B}$ means that $\mathcal{A}$ is submodel of $\mathcal{B}$ [CK73].

The proof of the following theorem is due to Dexter Kozen.

The statement and proof carry the implicit assumption that graphs are represented in the usual manner; i.e., as finite structures $G = (V, E)$ where V is the set of vertices and E is the edge predicate. We remark on other representations after the proof.

Theorem 2 MAXCLIQUE *is not in* MAX NP.

PROOF. The proof is by contradiction. Assume that MAXCLIQUE belongs to MAX NP, meaning that for all graphs G,

$$opt_{CLQ}(G) = \max_S \| \{\overline{x} \mid \exists \overline{y}\ \Phi(\overline{x}, \overline{y}, E, S)\} \|$$

where Φ is quantifier-free, and for some fixed $rs, t > 0$, $\bar{x} = (x_1, ..., x_t)$, $\bar{y} = (y_1, ..., y_r)$, and $S \subseteq V^s$. Let us consider one particular $G_1 = (V_1, E_1)$ (with the only requirement that it has nonempty edge set) and let S_1 be such that

$$opt_{CLQ}(G_1) \;=\; \| \{\overline{x} \mid \exists \overline{y}\ \Phi(\overline{x}, \overline{y}, E_1, S_1)\} \|$$

Now we construct a new graph G_{new}. Let $G_2 = (V_2, E_2)$ be an isomorphic copy of G_1 and let $G_{new} = (V_1 \cup V_2, E_1 \cup E_2)$, i.e. there are no edges between G_1 and G_2. Hence, $opt_{CLQ}(G_{new}) = opt_{CLQ}(G_1) = opt_{CLQ}(G_2) \stackrel{def}{=}$ OLDVALUE We claim that,

$$\max_S \| \{\overline{x} \mid \exists \overline{y}\ \Phi(\overline{x}, \overline{y}, E_{new}, S)\} \| \;\geq\; 2 \cdot \text{OLDVALUE}$$

Given a tuple $\overline{x}$ of vertices in G_1 we will indicate with $\overline{x}'$ the tuple made of the corresponding elements in G_2. Similarly, S_2 is the "isomorphic copy" of S_1; that is

$$< a_1, ..., a_s > \in S_1 \;\Leftrightarrow\; < a'_1, ..., a'_s > \in S_2$$

We have that

$$\exists \overline{y}\ \Phi(\overline{a}, \overline{y}, E_1, S_1) \ \Leftrightarrow \ \exists \overline{y}\ \Phi(\overline{a}', \overline{y}, E_2, S_2)$$

Choose $S_{new} = S_1 \cup S_2$. We make the subclaim that, for all $\bar{a} \in V_1^t$,

$$\exists \overline{y}\ \Phi(\overline{a}, \overline{y}, E_1, S_1) \ \Rightarrow \ \exists \overline{y}\ \Phi(\overline{a}, \overline{y}, E_{new}, S_{new}).$$

To see this, assume $\bar{a}$ to be such that $G_1, S_1 \models \exists \overline{y}\ \Phi(\overline{a}, \overline{y}, E, S)$. This implies that there exists $\bar{b}$ such that $\Phi(\overline{a}, \overline{b}, E_1, S_1)$ is true. We will show that $\Phi(\overline{a}, \overline{b}, E_{new}, S_{new})$ is also true. We will show that the truth values of the atoms of Φ in the two cases are the same. The atoms of $\Phi(\overline{a}, \overline{b}, E_{new}, S_{new})$ are of the form $E_{new}(\bar{z})$, $S_{new}(\bar{w})$, where $\bar{z}$ and $\bar{w}$ are tuples of elements taken from the set $\{a_1, ..., a_t, b_1, ..., b_r\}$, or of the form $x = y$ where x and y range over $\{a_1, ..., a_t, b_1, ..., b_r\}$. But then, since $S_{new} = S_1 \cup S_2$ and $E_{new} = E_1 \cup E_2$,

$$E_{new}(\bar{w}) \Leftrightarrow E_1(\bar{w}) \quad \text{and} \quad S_{new}(\bar{z}) \Leftrightarrow S_1(\bar{z}).$$

A simple structural induction on formulae then shows:

$$\Phi(\overline{a}, \overline{b}, E_1, S_1) \Leftrightarrow \Phi(\overline{a}, \overline{b}, E_{new}, S_{new})$$

which proves the subclaim. Similarly,

$$\exists \overline{y}\ \Phi(\overline{x}', \overline{y}, E_2, S_2) \ \Rightarrow \ \exists \overline{y}\ \Phi(\overline{x}', \overline{y}, E_{new}, S_{new})$$

,

and hence

$$\|\{\overline{x} \mid \exists \overline{y}\ \Phi(\overline{x}, \overline{y}, E_{new}, S_{new})\}\| \ \geq \ 2 \cdot \text{OLDVALUE}$$

because $\{\bar{x} \mid \exists \overline{y}\ \Phi(\overline{x}, \overline{y}, E_1, S_1)\}$ and $\{\bar{x}' \mid \exists \overline{y}\ \Phi(\overline{x}', \overline{y}, E_2, S_2)\}$ are disjoint. $\square$

The theorem was proved under the assumption that a graph is a finite structure of the kind $G = (V, E)$. However, what we actually used in the proof were the following assumptions on the coding of graphs via finite structures. First, isomorphic graphs are represented by isomorphic structures and isomorphic structures represent isomorphic graphs. Second, if $G_1 = (V_1, E_1)$ has a coding $G_1 = (A_1, P_1^1, .., P_m^1)$ and $G_2 = (V_2, E_2)$ has a coding $G_2 = (A_2, P_1^2, .., P_m^2)$ then $G = (V_1 \cup V_2, E_1 \cup E_2)$ has a coding isomorphic to $G = (A_1 \cup A_2, P_1^1 \cup P_1^2, .., P_n^1 \cup P_n^2)$. These conditions are satisfied by any reasonable encoding of graphs.

We now introduce a family of optimization problems. We first show that they are approximable and then that they do not belong to MAX NP.

This family is a natural generalization of the MAXIMUM MATCHING problem.

Suppose we are given a set of k-tuples $T = \{T_1, .., T_n\} \subseteq A_1 \times A_2 \times ... \times A_k$, where the A_i's are pairwise disjoint sets. Say two tuples are *compatible* if they differ in all k components. Then a set $M \subseteq T$ is a *matching* if every two k-tuples in M are compatible.

MAX k-DIMENSIONAL MATCHING (MAX k-DM).

INSTANCE. A collection of k-tuples $T = \{T_1, .., T_n\}$.

PROBLEM. Find the maximum size matching.

When $k = 2$, MAX k-DM is equivalent to the MAXIMUM MATCHING problem on bipartite graphs that is known to be in P. MAX 3DM is the optimization version of the NP-complete problem 3DM [GJ79].

Proposition 3 *For all* $k \geq 2$, MAX k-DM *is in* APX.

PROOF. One can show that the size of any *maximal* matching is at least $1/k$ of the size of a maximum matching. □

The next theorem shows that MAX NP does not include all polynomially computable optimization problems.

Theorem 3 MAX 2DM *is not in* MAX NP.

PROOF. The proof is similar to that of Theorem 2. Assume, for contradiction, that MAX 2DM $\in$ MAX NP. Then, there exists a formula Φ such that for all instances I of MAX 2DM,

$$opt_{2DM}(I) = \max_S \| \{\overline{x} \mid \exists \overline{y}\ \Phi(\overline{x}, \overline{y}, I, S)\} \|.$$

Consider an instance $I_1 = \{T_1, .., T_n\}$ such that $opt_{2DM}(I_1) = n$, i.e. I_1 is a set of n pairwise compatible pairs (we can look at I_1 as a collection of n disjoint edges).

From our assumption for contradiction, we have that there is S_1 such that

$$opt_{2DM}(I_1) \ = \ \| \{\overline{x} \mid \exists \overline{y}\ \Phi(\overline{x}, \overline{y}, I_1, S_1)\} \| = n.$$

Let $\overline{x}_1, .., \overline{x}_n$ be the tuples satisfying the above formula. Consider $\overline{x}_1$ and suppose, without loss of generality, that it contains a_1, i.e. $\overline{x}_1 = (a_1, u_2, .., u_k)$, and that $T_1 = (a_1, b_1)$.

We now construct another instance I_2 by simply replacing a_1 with a brand new element a_0. Let $I_2 = \{T_0, T_2, .., T_n\}$ where $T_0 = (a_0, b_1)$. I_2 is made of the same tuples of I_1 except the first, T_0. T_0 and T_1 only differ for the first component, namely a_1. We choose a_0 so that I_2 is made of n mutually compatible tuples. Now define S_2 to be the same set as S_1 provided any occurrence of a_1 is replaced by an occurrence of a_0, and define $\bar{z}_i$ to be the same tuple as $\bar{x}_i$ provided the same substitution takes place. Then,

$$\| \{\overline{z} \mid \exists \overline{y}\ \Phi(\overline{z}, \overline{y}, I_2, S_2)\} \| = n.$$

If we now consider the new instance $I_{new} = I_1 \cup I_2$ and define $S_{new} = S_1 \cup S_2$, we have that $opt_{2DM}(I_{new}) = n$ but

$$\max_{S} \|\{\overline{w} \mid \exists \overline{y}\ \Phi(\overline{w}, \overline{y}, I_{new}, S)\}\| \ \geq\ \|\{\overline{w} \mid \exists \overline{y}\ \Phi(\overline{w}, \overline{y}, I_{new}, S_{new})\}\| \ \geq\ n+1$$

because

$$\|\{\overline{x}_1, .., \overline{x}_n\} \ \cup\ \{\overline{z}_1, .., \overline{z}_n\}\| \geq n+1.$$

This contradiction shows that MAX 2DM $\not\in$ MAX NP. □

Basically the same proof applies to MAX k-DM, for all $k \geq 2$.

Corollary 1 *For all $k \geq 2$,* MAX k-DM *does not belong to* MAX NP.

We now introduce another family of problems, similar to MAX k-DM. Given a collection of sets of cardinality k, $S = \{S_1, .., S_n\}$, we define a *packing* to be a collection of pairwise disjoint sets: $S_i \in C \ \wedge\ S_j \in C \ \Rightarrow\ S_i \cap S_j = \emptyset$.

MAX K-SET PACKING (MAX k-SP).

INSTANCE. A collection $S = \{S_1, .., S_n\}$ of sets, where each S_i has cardinality k.

PROBLEM. Find a packing of maximum size.

MAX k-SP is the natural optimization version of the problem SET PACKING [GJ79]. We claim, without proof, that the following theorems hold. Their proofs are very similar to the theorems we saw for MAX k-DM.

Theorem 4 *For all $k \geq 2$,* MAX k-SP *is in* APX.

Theorem 5 *For all $k \geq 2$,* MAX k-SP *does not belong to* MAX NP.

All the problems we introduced in this section fit nicely in a new complexity class.

Definition 12 MAX Π_1 *is the class of* NP *optimization problems F such that, for all input instances I,*

$$opt_F(I) = \max_S \| \{ \overline{x} \mid \forall \overline{y}\, \Phi(G, S, \overline{x}, \overline{y}) \} \|.$$

As an example, consider MAXCLIQUE. It is easy to see that, for all graphs G,

$$opt_{CLQ}(G) = \max_C \| \{ x \mid C(x) \ \wedge \ \forall yz \ (C(y) \wedge C(z) \rightarrow E(y,z) \vee y = z) \} \|$$

where x, y, and z range over vertices and $E(y,z) = $ TRUE iff $(y,z) \in E$.

The proposition which we state next has a trivial proof, which is omitted.

Proposition 4 MAXCLIQUE, MAX k-DM, MAX k-SP *belong to* MAX Π_1.

MAX Π_1 is a natural way of expressing many NPO problems. In the next section we will prove completeness for natural variants of SAT.

In particular, our canonical complete problem will be the following.

MAX NUMBER OF ONES (MAX ONES).

INSTANCE. A boolean formula φ in *3CNF* .

PROBLEM. Find a satisfying assignment with the maximum number of variables set to TRUE.

We can express MAX ONES as a MAX Π_1 problem as follows. As in the case of 3SAT the instance is coded by means of four predicates $C_0, .., C_3$ where $C_i(x,y,z) =$ TRUE iff φ has a clause whose variables are x, y, and z and where the first i among its variables appear negated (e.g. $C_2(x,y,z)$ means $(\bar{x} \vee \bar{y} \vee z)$ is a clause) [PY88]. Then,

$$opt_{ONES}(\varphi) = \max_T \| \{ x \mid T(x) \wedge \forall yzw \ \Phi(\varphi, T, x, y, z, w) \} \|$$

where

$$\begin{aligned}\Phi(\varphi, T, x, y, z, w) &= (C_0(y,z,w) \rightarrow T(y) \vee T(z) \vee T(w)) \wedge \\ &\quad (C_1(y,z,w) \rightarrow \neg T(y) \vee T(z) \vee T(w)) \wedge \\ &\quad (C_2(y,z,w) \rightarrow \neg T(y) \vee \neg T(z) \vee T(w)) \wedge \\ &\quad (C_3(y,z,w) \rightarrow \neg T(y) \vee \neg T(z) \vee \neg T(w))\end{aligned}$$

3.5 Structural Properties of MAX Π_1

In this section we exhibit complete problems for the class MAX Π_1 . We also show that the complete problems for the class are non-approximable unless P = NP. Our first MAX Π_1-complete problem is the following.

MAX NUMBER OF SATISFIABLE FORMULAE (MAX NSF).

> INSTANCE. A set of *3CNF* formulae $\{\varphi_1, \varphi_2, ..., \varphi_n\}$.
>
> PROBLEM. Find a truth assignment to the variables such that the maximum number of the formulae are satisfied.

In this problem, the set of feasible solutions of non zero weight are the assignments satisfying at least one formula φ_i; this implies that approximating MAX NSF is NP-hard.

Theorem 6 MAX NSF *is* MAX Π_1*-complete under A-reductions.*

PROOF. We first show that MAX NSF $\in$ MAX Π_1. Informally, this is because the optimum value on instance I can be expressed as

$$opt_{\text{MAX NSF}}(I) = \max_T \|\{i \mid \varphi_i(T) = \text{TRUE},\ 1 \le i \le n\}\|$$

where I is the input instance $\{\varphi_1, \varphi_2, ..., \varphi_n\}$ and T is a unary predicate which is basically a truth assignment to the variables in $\{\varphi_1, \varphi_2, ..., \varphi_n\}$.

To write this formally, we may suppose that I is presented via two 3-ary predicates P and N where $P(i,j,k)$ is true iff variable x_k occurs positively in the jth clause of the formula φ_i , C_{ij}, and $N(i,j,k)$ is true iff variable x_k occurs negatively in C_{ij}. Then, more precisely,

$$opt_{\text{MAX NSF}}(I) = \max_T \|\{i \mid \forall j k_1 k_2 k_3 \; \Phi(I,T,i,j,k_1,k_2,k_3)\}\|$$

where

$$\begin{aligned}
\Phi \;=\; & (P(i,j,k_1) \wedge P(i,j,k_2) \wedge P(i,j,k_3) \rightarrow T(k_1) \vee T(k_2) \vee T(k_3)) \;\wedge \\
& (P(i,j,k_1) \wedge P(i,j,k_2) \wedge N(i,j,k_3) \rightarrow T(k_1) \vee T(k_2) \vee \neg T(k_3)) \;\wedge \\
& (P(i,j,k_1) \wedge N(i,j,k_2) \wedge N(i,j,k_3) \rightarrow T(k_1) \vee \neg T(k_2) \vee \neg T(k_3)) \;\wedge \\
& (N(i,j,k_1) \wedge N(i,j,k_2) \wedge N(i,j,k_3) \rightarrow \neg T(k_1) \vee \neg T(k_2) \vee \neg T(k_3)).
\end{aligned}$$

Second we establish the completeness of MAX NSF .

Let F be any optimization problem in MAX Π_1, and let f_F be its optimization function. Then

$$opt_F(I) = max_S \|\{\bar{x} \mid \forall \bar{y} \; \Psi(\bar{x},\bar{y},I,S)\}\|.$$

Recall that $\bar{x},\bar{y}$ represent fixed-arity tuples of variables. Hence, each tuple ranges over a polynomially sized domain (in the size of I). Let us enumerate the domain of x as $\bar{a_1},\bar{a_2},..,\bar{a_m}$ and the domain of y as $\bar{b_1},\bar{b_2},..,\bar{b_p}$. Each $\bar{a_i}$ is a tuple of names for elements of the domain, which can be substituted for for the respective variables of $\bar{x}$ in Ψ; similarly, the names $\bar{b_j}$ can be substituted for $\bar{y}$. Then for each i, $1 \leq i \leq m$, define φ_i to be the formula $\bigwedge_{1 \leq j \leq p} \Psi(\bar{a}_i,\bar{b}_j,I,S)$. Each φ_i is a polynomially sized boolean formula whose variables are $S(v_1,..,v_l)$ where S is an l-ary predicate. Moreover there are exactly m of these formulae. Since the formula Ψ is fixed in terms of F, the time taken to put Ψ itself into *CNF* is immaterial. Then, with the introduction of new variables, Ψ can be changed into a *3CNF* formula maintaining satisfiability. Hence, we can assume that each φ_i is a *3CNF* formula.

Now observe that, for any predicate assignment S_0 to S, the corresponding truth assignment S' to $\{S(v_1, ..v_l) | \, (v_1, ..v_l) \in domain(S)\}$ given by

$$S(v_1, \ldots v_l) = \text{TRUE} \;\Leftrightarrow\; (v_1, \ldots v_l) \in S_0$$

makes k-many formulae φ_i true iff $f_F(S_0) = k$. Hence this is an A-reduction, which concludes the proof. $\square$

We now show the MAX Π_1-completeness of MAX ONES with respect to P-reductions. We have already shown at the end of section 3.4 that MAX ONES is in MAX Π_1. To show hardness, we first exhibit a reduction from MAX NSF into an intermediate problem, MAX DONES, and then reduce MAX DONES to MAX ONES. MAX DONES is the following problem.

MAX DISTINGUISHED ONES (MAX DONES).

> INSTANCE. A boolean formula $\varphi(X, Z)$ where $X = \{x_1, .., x_n\}$ and $Z = \{z_1, .., z_m\}$. The z_i's are the *distinguished* variables.
>
> PROBLEM. Find a satisfying assignment for φ with the maximum number of distinguished variables set to TRUE.

Lemma 1 MAX DONES *is* MAX Π_1*-complete with respect to A-reductions.*

PROOF. MAX DONES can be written down as a MAX Π_1 problem in essentially the same way we wrote MAX ONES; besides the predicates C_i we need a predicate $D(z)$ that is TRUE iff z is a distinguished variable.

We reduce MAX NSF to MAX DONES. Given instance $\psi = \{\varphi_1(X), .., \varphi_n(X)\}$ of MAX NSF we construct the formula

$$F(X,Z) = (\varphi_1 \vee \neg z_1) \wedge ... \wedge (\varphi_n \vee \neg z_n)$$

By distributing the z_i's over the clauses of φ_i we can see that F is a 4CNF formula that satisfies the following: there is an assignment that makes k formulae $\varphi_{i_1}, ..., \varphi_{i_k}$ in ψ true if and only if there is a satisfying assignment for $F(X,Z)$ that sets $z_{i_1}, ..., z_{i_k}$ to TRUE.

This is an A-reduction. To complete the proof we have to transform $F(X,Z)$ into a *3CNF* formula. This can be done by introducing extra undistinguished variables y_i's; a clause $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is mapped into the two clauses $(x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee x_4)$. Since the y's are non-distinguished, this is again an A-reduction. □

Theorem 7 MAX ONES *is* MAX Π_1*-complete with respect to P-reductions.*

PROOF. We have already established that MAX ONES $\in$ MAX Π_1 at the end of the preceding section. To prove completeness, we transform MAX DONES into MAX ONES. Let $\varphi(X,Z)$ with $Z = \{z_1, ..., z_p\}$ and $X = \{x_1, .., x_q\}$ be an instance of MAX DONES; we transform it into an instance $\psi(X,Y,Z,Z')$ of MAX ONES. In what follows we will indicate with $\tau' : X \cup Y \cup Z \cup Z' \to \{0,1\}$ a satisfying assignment for ψ, and with τ the restriction to $X \cup Z$ of τ'. The reduction we are going to show is such that $\psi(\tau'(X), \tau'(Y), \tau'(Z), \tau'(Z')) = \text{TRUE} \Leftrightarrow \varphi(\tau(X,Z)) = \varphi(\tau'(X), \tau'(Z)) = \text{TRUE}$.

The instance of MAX ONES is the following formula

$$\psi(X,Y,Z,Z') = \varphi(X,Z) \wedge \beta(Z,Z') \wedge \alpha(X,Y,Z).$$

Y and Z' are sets of brand new variables while X and Z are the same variables appearing in φ.

In ψ any true variable contributes to the weight of a satisfying assignment τ'. We would like the contribution of the x's and y's to be negligible with respect to that of the z's. The mission of the subformula $\beta(Z, Z')$ is to amplify the weight carried by the variables z's. We define

$$\beta(Z, Z') = \bigwedge_{1 \le i \le p} \left(\bigwedge_{1 \le j \le 2l-1} z_i \leftrightarrow z_{ij} \right)$$

where $Z = \{z_1, .., z_p\}$ and $Z' = \{z_{ij} \mid 1 \le i \le p,\ 1 \le j \le 2l-1\}$. The index l is set to $l = q + r$, where r is the number of y's in ψ. l is selected after the construction of α is done. What β does is equivalent to assigning a weight of $2l$ to each z_i. Notice that β can be expressed in *CNF* with clauses of two literals. Also notice that any satisfying assignment for $\varphi(X, Z)$ automatically determines a satisfying assignment for $\beta(Z, Z')$ and that β can be expressed in *CNF* using $4lp$-many clauses of two literals each.

The mission of $\alpha(X, Y, Z)$ is to forbid truth assignments of ψ where some of the x_i's are set to true and all the z_i's are set to false. If this happened, we would have a solution of ψ with cost greater than zero mapped into a solution of φ of cost zero, i.e. approximated solutions would not be mapped into approximated solutions (recall that our transformation simply considers restrictions $\tau(X, Z) = \tau'(X, Z)$, where τ' satisfies ψ).

A way of implementing α would be to write down

$$\alpha = \bigwedge_{1 \le i \le q} \left(\neg x_i \vee \bigvee_{1 \le j \le q} z_j \right).$$

But these are clauses of unbounded length. To have clauses of length at most three, we transform each clause $(\neg x_i \vee z_1 \vee .. \vee z_q)$ into

$$(\neg x_i \vee z_1 \vee y_1) \wedge (\neg y_1 \vee z_2 \vee y_2) \wedge ... \wedge (\neg y_{q-1} \vee z_q).$$

It can be checked that $\alpha(X, Y, Z)$ so defined satisfies the two properties: *i)* the truth of any y_i or x_i implies the truth of some z_j; *ii)* any truth assignment τ' of $\varphi(Z, X)$ can be extended to a truth assignment $\tau \supseteq \tau'$ satisfying $\alpha(X, Y, Z)$.

To summarize, $\psi(X,Y,Z,Z')$ can be expressed in *3CNF* and the restrictions to $X \cup Z$ of its satisfying assignments form the set of satisfying assignments for $\varphi(X,Z)$.

We now have to show that the reduction is a P-reduction. The transformation can certainly be carried over in polynomial time.

Let $opt_{DONES}(\varphi) = k$. By construction it follows that $opt_{ONES}(\psi) \geq 2lk$. It also follows that the possible weights for a solution τ' of ψ are $w(\tau') = 0, 2l + n_1, .., 2li + n_i, ..2lk + n_k$ where $1 \leq i \leq k$ and $n_i \leq l = q + r$ for all i's. Moreover, the relationship between a solution τ' and its restriction τ is given by $w(\tau') = 2li + n_i \Leftrightarrow w(\tau) = i$ and $w(\tau') = 0 \Leftrightarrow w(\tau) = 0$.

We want to show

$$\frac{opt(\psi) - w(\tau')}{opt(\psi)} \leq \frac{\epsilon}{2} \Rightarrow \frac{opt(\varphi) - w(\tau)}{opt(\varphi)} \leq \epsilon$$

In order to do so, it is enough to prove

$$\frac{opt(\psi) - w(\tau')}{opt(\psi)} \geq \frac{opt(\varphi) - w(\tau)}{2\ opt(\varphi)}$$

Consider a solution τ' of ψ. When $w(\tau') = 0$ or $w(\tau') \geq 2lk$ the above equation holds. Suppose then that $w(\tau') = 2li + n_i$ with $1 \leq i \leq k - 1$. We have

$$\begin{aligned}
\frac{opt(\psi) - w(\tau')}{opt(\psi)} &\geq \frac{2lk - w(\tau')}{2lk} && (3.1)\\
&\geq \frac{2lk - l(2i+1)}{2lk} && (3.2)\\
&= \frac{2k - (2i+1)}{2k}\\
&\geq \frac{k-i}{2k}\\
&= \frac{opt(\varphi) - w(\tau)}{2\ opt(\varphi)}
\end{aligned}$$

Equation (1) holds since $opt(\psi) \geq 2lk$, while equation (2) holds since $w(\tau') = 2li + n_i \leq l(2i+1)$. This concludes the proof. $\square$

The complete problems we saw are not approximable unless P = NP. However, we know that MAX Π_1 contains approximable problems, like MAX k-DM, and problems that are believed not to be approximable like MAXCLIQUE, and which have the interesting property that they are either not approximable or are in PTAS. It would be interesting to characterize, within MAX Π_1, classes whose problems share similar approximation properties. In the next section, we see how it is possible to describe problems like MAX CLIQUE, MAX k-SP, and MAX k-DM by posing syntactic restrictions on the formulae Φ_F certifying membership of F in MAX Π_1.

3.6 Expressive Power of Restrictions of MAX Π_1

In the previous section we saw that MAX Π_1 in its full generality has problems which are too hard for approximation. On the other hand, let us examine the expressions for the optimization functions for various problems we have been discussing, and which we proved are not in MAX NP

- MAX CLIQUE. $opt_{CLQ}(G) = max_C \|\{x \mid C(x) \wedge \forall uv\ \Phi(C,E,u,v)\}\|$ where $\Phi(C,E,u,v) = (\neg C(u) \vee \neg C(v) \vee E(u,v))$.

- MAX 3DM. $opt_{3DM}(I) = max_M \|\{\bar{a} \mid M(\bar{a}) \wedge \forall \bar{b}\bar{c}\ \Phi(M,T,\bar{b},\bar{c})\}\|$ where $\Phi(M,T,\bar{b},\bar{c}) = [\neg M(\bar{b}) \vee T(\bar{b})] \wedge [\neg M(\bar{b}) \vee \neg M(\bar{c}) \vee \wedge_{i=1,2,3}(b_i \neq c_i)]$.

  Here $\bar{a}$ stands for (a_1, a_2, a_3) and I stands for the input instance (A,T) where $T \subseteq A^3$ with $T(\bar{a}) =$ TRUE iff *"$\bar{a}$ is a triple"*.

These problems are not only in MAX Π_1, but the fashion in which they are expressed is also rather similar. More precisely, all these problems can be expressed as:

$$opt_F(I) = \max_S\{\|S\| \ : \ \forall \bar{y}\ \Phi_F(\bar{y}, I, S)\}$$

where $\| S \|$ denotes $\| \{\bar{x} : S(\bar{x})\} \|$, $\bar{y}$ is a first-order variable and Φ_F is quantifier-free. Most importantly, if Φ_F is expressed in CNF then all occurrences of S occur negatively.

Definition 13 *A problem $F \in$ RMAX(k) if its optimization function can be expressed as*

$$opt_F(I) = max_S\{\|S\| \ : \ \forall \bar{y} \ \Phi(\bar{y}, I, S)\}$$

where Φ is a quantifier-free CNF formula with all occurrences of S in Φ being negative, S a single predicate appearing at most k times in each clause, and $\|S\|$ denotes $\|\{\bar{x} : S(\bar{x})\}\|$.

Definition 14 RMAX $= \bigcup_k$ RMAX(k).

This subclass may seem very restricted in the beginning, but it captures many of the problems in MAX Π_1 which are provably not in MAX NP. In fact, most of the problems we have considered are in RMAX(2) . Other problems which fall into this class include :

- MAX SP: this is a generalization of MAX k-SP. Given a collection $S_1, \ldots S_n$ of finite sets, find a packing of maximum size. This problem and MAX k-SP are in RMAX(2). Notice, MAX SP $= \bigcup_n$ MAX k-SP. Similarly, MAX DM and MAX k-DM are in RMAX(2).
- MAX INDEPENDENT SET: given a graph, find the size of the maximum independent set. This problem is in RMAX(2).
- MAX GRAPH k–COLORING: given a graph $G = (V, E)$ and an integer k, find the maximum number of vertices of G that can be colored with k colors such that no two adjacent vertices have the same color. This problem is in RMAX(2).

- MAX k-ANLSAT: this is the restriction of MAX ONES where all the variables in the input formula appear only negatively, and where every clause has at most k literals. This problem is in RMAX(k).

- MAX k-HYPERCLIQUE: An input instance is a k-hypergraph $H = (A, E)$ where A is a set and $E \subset \mathcal{P}(\mathcal{A})$ and $e \in E \Rightarrow 1 \leq |e| \leq k$. An element of E is called a hyperedge. A feasible solution is any set $W \subset A$ satisfying $\{u_1, \ldots, u_i\} \subseteq W \Rightarrow \{u_1, \ldots, u_i\} \in E,\ i \leq k$. Such a set is called a *k-hyperclique.* The goal is to find a k-hyperclique of maximum size.

  This problem is a generalization of the CLIQUE problem for graphs to hypergraphs and it is a trivial fact that MAX CLIQUE $\equiv_A$ MAX 2-HYPERCLIQUE.

Thus there is a large class of problems which are in RMAX(k). We now establish two theorems. The first theorem shows that MAX k-ANLSAT is complete in RMAX(k) with respect to A-reductions. The second theorem is about the equivalence of the families of problems MAX k-HYPERCLIQUE and MAX k-ANLSAT.

Theorem 8 *If $F \in$ RMAX(k) then $F \leq_A$* MAX k-ANLSAT.

PROOF. If $F \in$ RMAX(k), it can be expressed with a formula Φ_F with k occurrences of the predicate S per clause and with all negative occurrences of S. The rest of the proof is similar to that of Theorem 6. $\square$

Theorem 9 *For all $k \geq 2$,* MAX k-HYPERCLIQUE $\equiv_A$ MAX k-ANLSAT.

PROOF. We first show that MAX k-HYPERCLIQUE $\leq_A$ MAX k-ANLSAT. Let $H = (A, E)$ be any k-hypergraph. Construct ϕ_H as follows. The variables of ϕ_H are $\{x_i | i \in A\}$. ϕ_H is a conjunction of all the clauses of the form $(\neg x_{i_1} \vee \neg x_{i_2}, \ldots \vee \neg x_{i_k})$ where $x_{i_j} \in var(\phi_H)$ and $\{i_1, i_2, \ldots i_k\}$ is not a hyperedge in E. The literals may be repeated within a clause in which case it is simplified. These are the only clauses of ϕ_H. It can now be checked that ϕ_H can be satisfied with $x_{i_1}, x_{i_2}, \ldots x_{i_l}$ all set to TRUE if and only if $\{i_1, i_2, \ldots i_l\}$ form a hyperclique in H.

To prove that MAX k-ANLSAT $\leq_A$ MAX k-HYPERCLIQUE, use the inverse mapping. □

The last two theorems have interesting consequences.

Theorem 10 *Problems* MAX CLIQUE, MAX GRAPH k-COLORING, MAX SP *and* MAX DM *are* RMAX(2)*-complete with respect to A-reductions.*

PROOF. The completeness of MAX CLIQUE follows from Theorem 8 and the fact that MAX CLIQUE $\equiv_A$ MAX 2-HYPERCLIQUE. The remaining reductions are easy to obtain; for example, MAX CLIQUE $\leq_A$ MAX GRAPH k-COLORING because an independent set can always be 1-colored. □

All the RMAX(2)-complete problems share a very interesting property: either they are not approximable or they are in PTAS. The reason why, for example, MAX CLIQUE shows this behavior is that given any graph G we can, in polynomial time, construct G' such that: *i)* $opt(G') = opt(G)^2$; *ii)* if $C' \subseteq G'$ is a clique of k vertices then we can find in polynomial time a clique $C \subset G$ of at least $\sqrt{k}$ vertices. This implies that if $|C'|/opt(x') \geq 1 - \epsilon$ then $|C|/opt(x) \geq \sqrt{1-\epsilon}$. Since $\lim_{n\to\infty}(1-\epsilon)^{1/2^n} = 1$, we can obtain in G any approximation we want by iterating the above construction [GJ76,PS82].

The following definition generalizes this kind of situation. Recall that f_F is the objective function of the NPO problem F.

Definition 15 *A problem $F \in$* NPO *is* self-improvable *if there is a P-reduction $r = (t_1, t_2, c)$ from F to itself and a function h such that:*

- $h : (0,1) \to (0,1)$, *h is monotone increasing, and* $\lim_{n\to\infty} h^n(z) = 1$.
- $$\frac{f_F(x,y)}{opt_F(x)} \geq h\left(\frac{f_F(x',y')}{opt_F(x')}\right)$$

  where $x' = t_1(x)$, $y' \in S_F(x')$, *and* $y = t_2(x,y')$.

If a problem is self-improvable, then it is either in NPO − APX or in PTAS. The reason is that we can apply the reduction n times to map x into $t_1^n(x)$; an error of ϵ in the solution of $t_1^n(x)$ corresponds to an error ϵ_n in the solution of x, where ϵ_n tends to 0 as n tends to infinity. This can be seen as follows; let $x_n = t_1^n(x_0)$ and $y_n \in S_F(x_n)$, from the above definition it follows that

$$\frac{f_F(x_0,y_0))}{opt_F(x_0)} \geq h^n\left(\frac{f_F(x_n,y_n)}{opt_F(x_n)}\right)$$

where y_0 is obtained by repeated applications of t_2 on appropriate arguments starting from x_{n-1} and y_n.

For example, MAX CLIQUE is self-improvable with $h(z) = z^{1/2}$ [GJ76,PS82].

Fact 1 *If F is A-equivalent to G and G is self-improvable, so is F.*

We then have the following corollary.

Corollary 2 *All the A-complete problems of Theorem 10 are self-improvable.*

Notice that these results are obtained without directly mapping these problems to themselves.

3.7 Conclusion

In this chapter, we investigated the relationship between the logical expressibility of NPO problems and their approximation properties. To summarize, we have first shown that class MAX NP is rather weak in its expressive power. We have then defined another class of NPO problems based on logical structure. For this class we have demonstrated complete problems; moreover we have obtained interesting subclasses where the complete problems have similar properties with respect to approximation and in addition they all have the property of self-improvability. We see this work as a step in the direction of developing a general framework for establishing a connection between logical structure of a problem and its approximation properties.

3.8 State of the Art

Following this work, Kolaitis and Thakur have shown in their work [KT90] that, based on expressibility in terms of quantifiers, the NPO problems can be classified exactly into five classes which are all different. They also study the minimization problems and show that, surprisingly, they behave differently from the maximization problems[KT91]. We also refer to Viggo Kann's Ph.D. thesis for work related to that of this chapter and also a nice compendium of the current status of the approximability and hardness of various optimization problems[Kan92].

Chapter 4

The CounterExample Computation Model[2]

4.1 Introduction

In this chapter and the next, we shall be studying the difficulty of solving NP-optimization problems, exactly and approximately, in the *counterexample model.* Before we go on further we would like to briefly describe the counterexample model and provide some motivation as to why it is important and interesting to study the NP-optimization problems in the context of this model.

4.1.1 Prelude

As stated previously, most NPO problems are NP-hard, one can not expect to compute an optimal solution efficiently. Therefore, for such problems one concentrates on designing efficient algorithms which are guaranteed to compute good approximations. In many instances, for example MinTSP, even this may be hard and one has to rely on the heuristic approach. Even though heuristics work well in practice, unfortunately there is no theoretical guarantee on their performance.

[2]joint work with Suresh Chari and Pankaj Rohatgi

Many heuristics are incremental in nature. An incremental algorithm for an optimization problem works by repeatedly improving the previously computed solutions. So, in an abstract setting, an incremental algorithm can be thought of as an *improver* of solutions. A classic example of an incremental algorithm is the simplex method for linear programming. Other well-known examples are the famous edge-exchange procedures for MINTSP which were presented by Lin [Lin65] and Lin and Kernighan [LK73]. A more general example is the class PLS (polynomial-time local search)[JPY88], where every problem has an associated incremental algorithm to compute the local optimum. The notion of incremental computation is quite fundamental and should be understood *per se*. We believe that a study of optimization and approximation of NPO problems with respect to their amenability to solution by heuristics and incremental algorithms will lead to a better understanding of their structure. This will also shed new light on the puzzling fact that different NPO problems exhibit starkly different behavior even when their decision versions are computationally the same (*p-isomorphic*).

The counterexample model provides an abstract framework to study incremental computation in an abstract setting. This model was first introduced by Krajíček *et al* [KPS90] and studied further in [RCR91] , although with a different motivation. A counterexample protocol is a two-party interactive protocol: there is an all-powerful teacher, T, and a student, S, with limited power (polynomial time). The goal of S is to compute an optimal solution to the given instance of the optimization problem. To this end he 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 the presented solution is in fact optimal, T accepts, and the protocol halts, else T disproves the claim by presenting a counterexample, *i.e.* a better solution and the interaction is repeated. The difficulty of a problem is measured by the number of counterexamples the best student requires to compute the optimal solution given the least cooperative teacher. To study approximation properties we require S to compute only an approximate solution rather than an optimal solution and the protocol halts when an approximate solution with the desired accuracy is computed.

It is worth noting that this computation is very similar to a computation by an incremental algorithm which gets a better solution on each iteration but with no guarantees on *how much* better the solution is. Specifically, assume that we can show that a particular optimization problem requires at least $k(n)$ counterexamples on some input of length n to compute the desired answer. This implies that for any incremental heuristic, which operates only on the basis that the solution obtained at each step is better than the previous one, there may be instances where it requires at least $k(n)$ improving steps. Hence, establishing lower bounds in this model corresponds to the worst case analysis of such heuristics. The case where such guarantees do exist can be modeled by imposing constraints on the improved solutions provided by the teacher. For example, one may study heuristics with *additive guarantees* by imposing the constraint that the new solution provided by the teacher is at least $f(n)$ better than the previous solution, for some function f.

The model is also interesting because it relates conjectures about the relative powers of theories of bounded arithmetic to those about this model which are purely computational in nature. For example, if one could prove a super-polynomial complexity for the traveling salesperson problem, MINTSP, in this model, it will also prove that $S_2^1 \neq T_2^1$ [Bus86,KPS90].

In this chapter, we will be concerned about computing the exact solutions to the problems. In the next chapter, we shall consider approximate solutions. This chapter is organized as follows. In the next section we briefly review the counterexample model, giving the necessary definitions and relevant results. In Section 4.3, we establish the difficulty of computing the lexicographically maximum satisfying assignment of boolean formulas in this model. The proof uses a result (Lemma 2) which is central to this chapter and the next, and interesting in its own right. It states that, under the assumption that the Polynomial Hierarchy(PH) does not collapse to its third level, not only is it hard for some polynomial time machine to compute a satisfying assignment for a boolean formula but there are formulas for which it is hard to find a *new* satisfying assignment even given a lot of *non-trivial* ones which it can not compute by itself. 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.

4.2 The Counterexample Model

An *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 it is allowed an arbitrary number of steps over the entire computation.

Definition 16 *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 *form a counterexample protocol for* $\mathcal{P}$ *which requires no more than* $f(n)$ *counterexamples on inputs of size* n*.*

Definition 17 $\mathcal{C}[f(n)]$ *is the class of all* NP*-optimization problems which have an* $f(n)$*-counterexample protocol.*

It is easy to see that MAXCLIQUE has an n-counterexample protocol. Given a graph G the student first presents a one vertex clique. Then the student repeatedly presents the same solution that 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 by Krajíček *et al* (1990) that

Theorem 11 *For all polynomial time constructible functions f such that $1 \leq f(n) \leq n^{1-\epsilon}$, $\mathcal{C}[f(n)] = \mathcal{C}[f(n)-1]$ implies* NP $\subseteq$ P/*poly, for all* $\epsilon > 0$.

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 12 *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 all* $\epsilon > 0$ there is no $n^{1-\epsilon}$-counterexample protocol for these problems. Since MAXCLIQUE has an 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.

4.3 The Complexity of Computing New Satisfying Assignments

In this section we establish lower bounds on the number of counterexamples required to compute the optimum solutions for LEXMAXSAT and MINTSP, under standard structural complexity theoretic assumptions. Towards this end we show a result about the problem SAT, which is interesting in its own right. The result states that there are polynomial sized collections of uniquely satisfiable formulas whose satisfying assignments are independent in the sense that knowledge of any subcollection of these does not convey information about any of the others. We then use these formulas to construct a single formula with many independent satisfying assignments and establish lower bounds for LEXMAXSAT in the counterexample model.

Notation 1 USAT *denotes the set of uniquely satisfiable formulas. For* $F \in$ USAT, $s(F)$ *denotes the unique satisfying assignment of* F. *For any set* A, $A^{=n}$ *denotes* $\{x \in A \mid |x| = n\}$.

We now formally define the notion of independence of a collection of formulas.

Definition 18 *Let* D *be a polynomial time transducer. A set* $F_1, \ldots, F_m$ *of formulas in* $\text{USAT}^{=n}$ *are said to be* independent *with respect to* D *if for all* j, $1 \leq j \leq m$, D *on input* $\{F_1, \ldots, F_m, s(F_1), \ldots, s(F_{j-1}), s(F_{j+1}), \ldots, s(F_m)\}$ *does not compute* $s(F_j)$.

A collection of formulas is dependent with respect to D, if it is not independent. The following lemma establishes that, for each transducer D, independent collections of formulas exist, unless USAT $\in$ co-NP/*poly*. We later show that if USAT $\in$ co-NP/*poly*, the Polynomial Hierarchy collapses.

Lemma 2 *If there exists a polynomial time transducer,* D, *and a polynomial* $r(\cdot)$ *such that, for all but finitely many* n, *every collection of* $r(n)$ *formulas in* $\text{USAT}^{=n}$ *is dependent with respect to* D, *then* USAT $\in$ co-NP/*poly*.

PROOF. Using D, we construct a co-NP machine, N, and a polynomial sized advice for each length, using which N recognizes USAT.

By assumption there exists n_0 such that for $n \geq n_0$, every collection $\mathcal{F} = \{F_1, \ldots, F_{r(n)}\}$ of formulas of length n is dependent with respect to D. Let $n > n_0$, and $r = r(n)$. Denote by $j(\mathcal{F})$ the least index j, between 1 and r, of the formula for which D computes $s(F_j)$ given $\mathcal{F} = \{F_1, \ldots, F_r\}$ and all $s(F_i)$ except $s(F_j)$. Finally, define the set help($\mathcal{F}$) as

$$\mathrm{help}(\mathcal{F}) = \mathcal{F} \setminus \{F_{j(\mathcal{F})}\}.$$

Note that for any collection $\mathcal{F}$, given the formulas in $\mathcal{F}$ and the satisfying assignments of the formulas in help($\mathcal{F}$), D computes the assignment satisfying the remaining formula $F_{j(\mathcal{F})}$. Consider any set of formulas Z, where $Z \subseteq \mathrm{U{\small SAT}}^{=n}$. For any collection $\mathcal{F} \subseteq Z$, of size r, help($\mathcal{F}$) is a set of size $r-1$. There are

$$\binom{|Z|}{r}$$

subsets of Z, of size r, and

$$\binom{|Z|}{r-1}$$

subsets of size $r-1$. By the pigeonhole principle, there must be some $r-1$ sized subset $\mathcal{F}'$ which is help($\mathcal{F}$) for at least

$$\binom{|Z|}{r} \Big/ \binom{|Z|}{r-1} = \frac{|Z| - r + 1}{r}$$

r sized subsets $\mathcal{F}$. Then, given this set $\mathcal{F}'$ and the solution of its formulas, D can compute the solution for at least $\frac{|Z|-r+1}{r}$ formulas. This follows from the observation that each $\mathcal{F}$ that maps onto $\mathcal{F}'$ gives a different formula whose satisfying assignment can be computed by D, given the formulas in $\mathcal{F}'$ and their satisfying assignments.

Thus for any set $Z \subseteq \mathrm{U{\small SAT}}^{=n}$ of size greater than $4r$, we can find a set of $r-1$ formulas, with which we can compute the satisfying assignment of at least $1/2r$ of the formulas in Z. We add this collection of formulas and their solutions to the advice. Repeating this process k times, as shown in Fig. 4.1, we can construct advice to recognize all but $|\mathrm{U{\small SAT}}^{=n}| \times (1 - 1/2r)^k$ strings in $\mathrm{U{\small SAT}}^{=n}$. If we choose $k = 4 \times n \times r$ at most polynomially many ($4r$) formulas remain. We add these and their solutions to the advice. Given this advice $\mathcal{A} = \mathcal{A}'_1\# \ldots \#\mathcal{A}'_k$ a co-NP machine, N, which works as follows, recognizes USAT.

Given a boolean formula F, of length n, as input

- If $n < n_0$, use a table lookup to decide if $F \in$ USAT.
- Check that F has no more than one satisfying assignment.
- If F and its satisfying assignment appear in the advice or for some j, D given $\mathcal{A}'_j$ and F computes $s(F)$ then accept.

Hence USAT $\in$ co-NP/*poly*.

The above lemma implies the existence of independent formulas under the assumption that USAT $\notin$ co-NP/*poly*. The following proposition shows that this is as reasonable as the assumption that the Polynomial Hierarchy does not collapse to its third level.

Proposition 5 *If* USAT $\in$ co-NP/*poly then* PH *collapses to* Σ_3^{P}.

PROOF. The proof of this proposition relies on the following two facts:

1. co-NP/*poly* is closed under disjunctive reductions.
2. SAT randomly reduces to USAT with a success probability of $1/4n$ [VV86].

If USAT $\in$ co-NP/*poly*, then by the second fact, SAT randomly reduces to some language in co-NP/*poly* with a success probability of $1/4n$. Using the first fact, by repeated trials, we can randomly reduce SAT to another language in co-NP/*poly* with a success probability of $1 - 2^{n^2}$. Then, using techniques due to Bennett *et al*[BG81], we can build advice such that SAT $\in$ (co-NP/*poly*)/*poly* = co-NP/*poly*. This implies that NP/*poly* = co-NP/*poly* and PH collapses to Σ_3^{P} [Yap83]. Under standard assumptions, the lemma shows that there are large collections of equal sized boolean formulas whose satisfying assignments are independent, in the sense that satisfying assignments of one provide no information about those of any

$Z \leftarrow \text{USAT}^{=n}$

$\mathcal{A} \leftarrow$ "null string" /* A is the advice string to be computed */

While $|Z| \geq 4r$ **do**

begin

find $\mathcal{F}' \subseteq Z$ of size $r-1$ such that $|\text{help}^{-1}(\mathcal{F}')| \geq (|Z|-r)/r$

Let $\mathcal{F}'=\{F_1, \ldots F_{r-1}\}$

/* One such $\mathcal{F}'$ exists by the proof */

$Z \leftarrow Z \setminus \text{help}^{-1}(\mathcal{F}')$

$\mathcal{A}' \leftarrow \mathcal{F}', s(F_1), \ldots, s(F_{r-1})$

$\mathcal{A} \leftarrow \mathcal{A}\#\mathcal{A}'\#$

end

$\mathcal{A} \leftarrow \mathcal{A}\#Z, s(Z_1), s(Z_2) \ldots s(Z_m)\#$

/* where $Z = \{Z_1, Z_2 \ldots Z_m\}$ */

Figure 4.1: Algorithm to construct advice for strings of length n

other. It is then possible to construct boolean formulas, with several satisfying assignments, for which the different satisfying assignments are independent 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 13 *If for some* $\epsilon > 0$, LEXMAXSAT $\in \mathcal{C}[n^{1-\epsilon}]$, *then the* PH *collapses to* Σ_3^P.

PROOF. Fix an $\epsilon > 0$ and assume that there is an $n^{1-\epsilon}$-counterexample protocol for LEXMAXSAT. Let D be the student in this protocol. Choose a polynomial $r(\cdot)$ such that $c \times (n \times r(n))^{1-\epsilon} < r(n)$, for any constant c. For instance, we can choose $r(n) = n^{\frac{1}{\epsilon}+1}$. Let $r = r(n)$. Define a polynomial time transducer D' which on input $\{F_1, \ldots F_r, y_1, \ldots y_{k-1}, y_{k+1} \ldots y_r\}$ does the following:

- Checks each y_i is a satisfying assignment of F_i.
- D' simulates D on $F = F_1 \bigvee \ldots \bigvee F_r$. Whenever D presents a solution y to the teacher, if y is a satisfying assignment of F_k then D' outputs y and stops. Otherwise D' continues the simulation assuming that the teacher provided the lexicographically smallest y_j which is larger than y.

Let $F_1, \ldots, F_r$ be a collection of formulas from USAT$^{=n}$. Consider the formula $F = F_1 \bigvee \ldots \bigvee F_r$ where the variables of all the F_i's are the same. The **only** satisfying assignments of F are $s(F_1), \ldots, s(F_r)$. Consider a teacher which provides counterexamples as follows: Whenever the student presents a solution y of F, it returns the lexicographically smallest $s(F_i)$ which is larger than y. The length of F is $c \times n \times r$ for some constant c. By assumption, D takes at most $(c \times n \times r)^{1-\epsilon}$ counterexamples to compute the largest satisfying assignment of F against all teachers. By choice of the polynomial $r(\cdot)$, $(c \times n \times r)^{1-\epsilon}$ is less that r for large n. This means that D, on input F, cannot follow the trivial strategy. Hence at some point of time in the protocol D computes a *new* satisfying assignment for F by itself. Suppose that the first time this happened, $s(F_i)$ was produced. Then, by the definition of D', $D'(F_1, F_2, \ldots, F_r, s(F_1), \ldots, s(F_{i-1}), s(F_{i+1}), \ldots, s(F_r))$ outputs $s(F_i)$. Thus the collection of formulas $F_1, \ldots, F_r$ is dependent with respect to D'.

Using the student D we have constructed a transducer D' such that every r sized collection is dependent with respect to D'. Hence by Lemma 2 and Proposition 5 the Polynomial Hierarchy collapses to its third level.

From the proof we can observe that, unless PH collapses, any student requires at least $r(n)$ counterexamples where n is the number of variables of F. Thus, for all polynomials $r(\cdot)$ there is no $r(n)$-counterexample protocol where n is the number of variables.

We can also observe the following.

Corollary 3 *If the* PH *does not collapse to its third level, in any counterexample protocol for* LEXMAXSAT, *the student can be forced to follow the trivial strategy for* $n^{1-\epsilon}$ *steps on infinitely many formulas, for all* $\epsilon > 0$.

We now consider the traveling salesperson problem MINTSP. Since the standard TSP is a minimization problem we define an equivalent maximization problem by defining the cost of a tour to be the sum of the weights of the edges in the graph minus the sum of the weights of the edges in the tour.
In order to establish lower bounds for MINTSP we use the language

$$\text{UNIQOPTTSP} = \{G \mid G \text{ has a unique Hamiltonian tour of minimum size}\}.$$

UNIQOPTTSP is known to be NP-hard [Pap84] Thus, if it is in co-NP/*poly* then the PH collapses.

Notation 2 *For any graph* G *in* UNIQOPTTSP, $t(G)$ *denotes its unique minimum tour.*

As in the case of LEXMAXSAT we say that a set of graphs $G_1, \ldots, G_m$ in UNIQOPTTSP$^{=n}$ are independent with respect to a transducer D if it does not compute $t(G_j)$ on input $\{\ G_1, \ldots, G_{r(n)}, t(G_1), \ldots, t(G_{j-1}), t(G_{j+1}), \ldots, t(G_{r(n)})\ \}$.

Lemma 3 *Assume that there exists a polynomial time transducer, D, and a polynomial $r(\cdot)$ such that, for all but finitely many n, every collection of $r(n)$ graphs in* UNIQOPTTSP$^{=n}$ *is dependent with respect to D. Then* UNIQOPTTSP *is in* co-NP*/poly and hence the* PH *collapses.*

PROOF. Similar to the proof of Lemma 2.

Using this we establish the desired lower bound on the number of counterexamples required to compute MINTSP.

Theorem 14 *If for some $\epsilon > 0$,* MINTSP $\in \mathcal{C}[n^{1-\epsilon}]$ *then the* PH *collapses.*

PROOF. The proof is similar to that of Theorem 13. A sketch is given here. 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. Choose a polynomial $r(\cdot)$such that $c \times (n \times r(n))^{1-\epsilon} < r(n)$ for any constant c. 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' constructs a new graph G from the input graphs $G_1, \ldots, G_{r(n)}$ and simulates D on G. Conceptually the new graph G can be thought of as a chain G_1–G_2–$\cdots$ –$G_{r(n)}$. The link between G_1 and G_2 is constructed as follows: Let $u \in G_i$ and $v \in G_{i+1}$ be two vertices which are not part of any other link. Substitute u with two copies of u, say u' and u'', in G_i, *i.e.*, 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 $|G| \leq c \times n \times r(n)$ 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 UNIQOPTTSP then so is G. In that case the weight of the optimal tour of G is the sum of the weights of the optimal tours of $G_i, 1 \leq i \leq n$.

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 by D 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 from among all solutions output by D in previous rounds of interaction.

Now exactly as in Theorem 13, if D computes the unique optimal tour for G in $n^{1-\epsilon}$ rounds then every collection of $r(n)$ graphs in $\text{UNIQOPTTSP}^{=n}$ is dependent with respect to D' and hence by Proposition 5, the Polynomial Hierarchy collapses.

4.4 Tight Bounds for polynomial valued problems

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 NPO problems which have trivial n-counterexample protocols and establish strong lower bounds on the number of counterexamples required in any protocol computing their optimal solutions. In particular we consider MAXCLIQUE, the problem MAXINDSET which is the problem of finding an independent set of vertices of maximum cardinality in a given graph, and the maximization version of the problem MINCOVER which is the problem of finding a vertex cover of minimum cardinality for a given graph.

In order to establish bounds for MAXINDSET we use the language

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

Since SAT can be parsimoniously reduced to the the independent set decision problem [GJ79], there is a reduction from USAT to UNIQ_INDSET. Hence if UNIQ_INDSET is in co-NP/*poly* then USAT $\in$ co-NP/*poly* and hence, by Proposition 5, PH collapses. As in the case of LEXMAXSAT we establish the following lemma to prove the result.

Notation 3 *For any graph G in* UNIQ_INDSET, $\imath(G)$ *denotes its unique maximum independent set. Also,* UNIQ_INDSET$^{=n}$ *denotes the set of all n-vertex graphs in* UNIQ_INDSET.

Lemma 4 *Assume that there exists a deterministic transducer D and a polynomial $r(\cdot)$ such that for all but finitely many n, for every set of graphs $G_1,\ldots,G_{r(n)}$ in* UNIQ_INDSET$^{=n}$, *there exists a j , $1\leq j \leq r(n)$, such that D when given the input $\{\ G_1,\ldots,G_{r(n)},\imath(G_1),\ldots,\imath(G_{j-1}),\imath(G_{j+1}),\ldots,\imath(G_{r(n)})\ \}$ and the size of $\imath(G_j)$, can compute $\imath(G_j)$. Then the* PH *collapses.*

PROOF. Suppose the hypothesis is true. We can construct, exactly as in Lemma 2, a deterministic machine D and a polynomially long advice string $S{=}S_1\#S_2\ldots\#S_k$, such that for all graphs G in UNIQ_INDSET$^{=n}$, D given G, the advice S and $d = |\imath(G)|$ can compute the maximum independent set of G. Here each S_i is the encoding of $r(n)-1$ n-vertex graphs and their unique maximum independent sets. Then the following co-NP machine accepts UNIQ_INDSET.

Given a graph G with n vertices as input

- For each value of d from 1 to n, run D on the input graph G, the advice S and assuming that $|\imath(G)| = d$. If $d = |\imath(G)|$ then D correctly computes the largest independent set of G. Let l be the size of the largest independent set for all values of d.

- Check that there is no more than one independent set of size l in G.

This implies that UNIQ_INDSET $\in$ co-NP/*poly* and hence, by Proposition 5, that the PH collapses.

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

Theorem 15 *If* for some $\epsilon > 0$, *there is a protocol for* MAXINDSET *that requires at most* $n^{1-\epsilon}$ *counterexamples for* n*-vertex graphs, then the* PH *collapses.*

PROOF. 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 $c(n \times r(n))^{1-\epsilon} < r(n)$. Define a transducer D' as follows:

On input $\{G_1, \ldots, G_{r(n)}, S_1, \ldots, S_{t-1}, S_{t+1}, \ldots, S_{r(n)}, k\}$, where each S_i is a set of vertices in the graph G_i and k is an integer

- Checks each S_i is an independent set of vertices in G_i.

- D' simulates D on G, the disjoint union of $G_1, \ldots, G_{r(n)}$. 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, 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 13, we can show that D' on input $G_1, \ldots, G_{r(n)}$ in UNIQ_INDSET and the solutions to all but G_j and the size of the largest independent set of G_j can compute $\imath(G_j)$.

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 surprising that this strategy is almost optimal.

4.5 Protocols with Randomness

So far, we have considered counterexample protocols in which the student is a deterministic polynomial time machine. In this section, we shall examine counterexample protocols in which the student also has access to a fair coin which can be tossed polynomially many times during the entire protocol. The resulting protocol is called a *probabilistic counterexample protocol.* The probabilistic protocol is similar to the deterministic counterexample protocol; the only difference being that, in addition to the problem instance, the student receives a polynomial sized string z which has been generated uniformly at random. The string z is provided on a separate tape and is not considered as part of the input. The student is a deterministic machine and as in the case of the deterministic protocol, repeatedly provides feasible solutions to the teacher. The teacher either accepts the given solution or provides a better solution as a counterexample. The student cannot take more than a polynomial amount of time between two successive interactions with the teacher. The protocol terminates when the teacher accepts. There is no restriction on the amount of time the entire protocol takes.

One could alternately define a different type of probabilistic counterexample protocol by allowing the student to be a polynomial time-bounded machine with access to a fair coin. In that case, the number of times the student is allowed to toss its coin during the entire protocol would depend on the number of interactions it has with the teacher. Thus, if the protocol takes exponentially many counterexamples for some input, then the student would be able to toss its coin exponentially many times. However, it is important to note that the two definitions are equivalent with respect to what a student can achieve in polynomially many interactions. Since we are only interested in what can be done using polynomially many interactions, we can use either definition. We choose the first definition because it simplifies the analysis.

Definition 19 *An optimization problem $\mathcal{Q}$ has a probabilistic $f(n)$-counterexample protocol if there is a student, S, and polynomials $s_p(\cdot)$ and $t_p(\cdot)$, such that, for all teachers T, S-T forms a probabilistic counterexample protocol for $\mathcal{Q}$ in which S, in addition to the input instance of size n, receives a random string of size $s_p(n)$. In addition, with probability $\geq 1/t_p(n)$, this protocol should require no more than $f(n)$-counterexamples.*

In the above definition, the probability is taken over the strings z of size $s_p(n)$ which are provided to S uniformly at random.

Definition 20 *$\mathcal{PC}[f(n)]$ 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 $1/n^c$. For any NP-optimization problem, there is always an $O(1)$ counterexample probabilistic protocol which works within the bound with inverse exponential probability. In this section we will show that if the student is required to work with "significantly" better probability (such as $1/n^c$), then there are are NP-optimization problems which do not have probabilistic $n^{1-\epsilon}$-counterexample protocols for any $\epsilon > 0$, unless the PH collapses. We first show that LEXMAXSAT is one such problem.

Lemma 5 *Suppose there is a probabilistic polynomial time transducer P and polynomials $r(\cdot)$ and $p(\cdot)$, such that for every collection $F_1, \ldots, F_{r(n)}$ in* USAT$^{=n}$*, there is a j such that P 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 compute $s(F_j)$ with probability $\geq 1/p(n)$. Then* USAT $\in$ BP $\cdot$ (co-NP/*poly*).

PROOF. The proof of this lemma is a straightforward extension of the proof of Lemma 2. In the proof Lemma 2, we showed how one can use the hypothesis of the lemma to build *advice* such that a polynomial time machine, using advice, could generate the satisfying assignment for any formula in USAT. In this case, one can similarly build advice, such that a randomized polynomial time machine, using the advice, can generate the satisfying assignment of any formula in USAT with probability $1/p(n)$. Using this machine, one can easily construct a randomized co-NP machine, which given advice, accepts formulas in USAT with high probability and always rejects formulas in $\overline{\text{USAT}}$.

The following proposition shows a consequence of the assumption that USAT is in BP $\cdot$ (co-NP/*poly*).

Proposition 6 *If* USAT $\in$ BP $\cdot$ (co-NP/*poly*) *then the Polynomial Hierarchy collapses.*

PROOF. The proof relies on the following facts

1. co-NP/*poly* is closed under majority reductions.

   PROOF. Given any language L define the languages:

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

   We have to show that for all $L \in$ co-NP/*poly*, MAJ$(L) \in$ co-NP/*poly*. It is well known that a language $L \in$ co-NP/*poly* iff $L \in$ co-NP^S for some sparse set S. Now if $L \in$ co-NP^S then $\overline{L} \in$ NP^S and, since NP^S is closed under majority reductions, MAJ$(\overline{L})$ is in NP^S. As MAJ$(\overline{L})$ = MIN(L) we have MIN$(L) \in$ NP^S . Thus $\overline{\text{MIN}(L)}$ = MAJ(L) is in co-NP^S and hence in co-NP/*poly*.

2. There is a random reduction which reduces SAT to USAT with a two sided error of at most $1/2 - 1/16n$ where n is the length of the string to which the reduction is applied.

PROOF. Valiant *et al* (1986) showed that there is a one-sided random reduction R which reduces SAT to USAT with probability $1/4n$. That is if $x \in$ SAT then Prob$[R(x) \in$ USAT$] \geq 1/4|x|$ and if $x \notin$ SAT then the probability that $R(x) \in \overline{\text{USAT}}$ is one.

Define the reduction R' as follows : On input x, $|x| = n$, R' outputs a trivial member of USAT with probability $1/2 - 1/16n$. With remaining $1/2 + 1/16n$ probability, R' simulates the reduction R on x. Thus if $x \in$ SAT then the probability that R' outputs a string in USAT is at least $(1/2 - 1/16n) + (1/2 + 1/16n)(1/4n)$ which is at least $(1/2 + 1/16n)$. If, on the other hand $x \notin$ SAT then, by definition of R, we have Prob$[R'(x) \notin$ USAT$] = 1/2 + 1/16n$.

Proof of Proposition: Assume USAT $\in$ BP $\cdot$ (co-NP/*poly*). By Fact 1 and Fact 2, this implies that SAT $\in$ BP $\cdot$ (co-NP/*poly*). Fact 1 also implies that BP $\cdot$ (co-NP/*poly*) = co-NP/*poly* (see Schöning 1989). Therefore SAT $\in$ co-NP/*poly* which implies that NP/*poly* = co-NP/*poly* and by a theorem of Yap(1983) PH collapses to Σ_3^{P}.

Using a proof similar to the deterministic case we can establish the following.

Theorem 16 *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 $1/t_p(n)$. Define the polynomial r and a transducer P' exactly as in Theorem 3. Let $q(n) = t_p(r(n)*n^2)$. From Lemma 4, for $p(n) = q(n) \times r(n) \times n$ there are infinitely many n's such that there are $r(n)$ formulae $F_1, F_2, \ldots F_{r(n)} \in \text{USAT}^{=n}$ 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 $F = F_1 \vee F_2 \ldots \vee F_{r(n)}$ where the variables of all the F_i's are the same. Let z be the random input to P where $|z| = s_p(|F|) = s(n)$ for some polynomial s. For every random input z to P, such that P works within its counterexample bound, P cannot follow the trivial strategy since $(|F|)^{1-\epsilon} < r(n)$. For each such z, 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 $1/(q(n) \times r(n))$ fraction of all z's of size $s(n)$. Therefore with probability $\geq 1/(q(n) \times r(n)) \geq 1/p(n)$, P' when given the input $(F_1, F_2, \ldots, F_{r(n)}, s(F_1), \ldots, s(F_{j-1}), s(F_{j+1}), \ldots, s(F_{r(n)}))$, outputs $s(F_j)$ contradicting Proposition 6.

Using a combination of techniques used in Lemmas 2 and 4 and Theorems 13, 15 and 16 we can also prove the following theorem :

Theorem 17 *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.*

Chapter 5

Approximation and Counterexample Model[3]

5.1 Introduction

We continue our investigation of the difficulty of the NP-optimization problems in the counterexample model. In this chapter we concern ourselves with approximating counterexample protocols - protocols where the student is required to compute only an approximately optimal solution instead of an optimal solution.

We examine the complexity of approximating representative NPO problems such as LEXMAXSAT and MAXCLIQUE in this model. Results about LEXMAXSAT carry over to many NPO problems such as MINTSP in which the cost function takes on exponentially many values. Similarly the results for MAXCLIQUE are applicable to many problems in which the optimal cost is polynomially bounded. One of the interesting result proved in this chapter pertains to the approximability of the much-studied NPO problem MAXCLIQUE[PS82,PR90,FGL+91,Blu91] and strengthens the breakthrough result by Feige *et al* [FGL+91] that shows that MAXCLIQUE cannot be approximated to within a factor of $f_1 = 1/2^{log^{1-\epsilon} n}$ in polynomial time, unless NP is in quasipolynomial ($2^{poly(\log n)}$) time.

[3]joint work with Suresh Chari and Pankaj Rohatgi

This chapter is organized as follows. In the next section we define approximating counterexample protocols and prove lower bounds on the number of counterexamples required to compute approximate solutions for representative NPO problems under standard structural complexity theoretic assumptions . Section 5.3 addresses the question: Are there any problems where a non-trivial strategy by the student will help in reducing the number of rounds of interaction? This question was first raised in [KPS90]. We present an example where using a non-trivial strategy to compute an approximate solution helps the student to reduce the number of rounds of interaction by a factor of $\log n$ over the trivial strategy. In section 5.4, we introduce protocols with guarantees. We again prove results which have the flavor of *"trivial is best"*.

5.2 Approximating protocols and Lower Bounds

Note that the standard counterexample protocol has two important features - firstly, the student is required to compute an optimal solution to the given instance, and secondly there are no guarantees on how much better the counterexample will be. In this chapter we are interested in studying the complexity of NPO problems, in terms of the number of counterexamples required, when we relax these conditions. In this section we consider the case when the student is only required to compute an approximate solution to the given instance. A protocol in which the teacher accepts when the student computes an α-approximate solution is called an α-approximating protocol.

Definition 21 *A solution s to an instance x of an* NPO *problem is α-approximate if the relative error of the solution, $\mathcal{E}(x, s) \leq 1 - \alpha$.*

For definition of relative error see chapter 2.

Definition 22 *An* NPO *problem $\mathcal{P}$ has an α-approximating $f(n)$-counterexample protocol if there is a student which computes an α-approximate solution using at most $f(n)$ counterexamples, on inputs of length n, against all possible teachers.*

In section 5.4 we modify the second aspect and insist that the teacher provide counterexamples which are guaranteed to be better than the student's solution by a predetermined amount.

In this section we establish lower bounds on the number of counterexamples required to compute approximate solutions for representative NP optimization problems, under usual structural complexity assumptions. Among exponential valued problems we choose LEXMAXSAT as a representative. Recall that LEXMAXSAT is the problem of computing the lexicographically largest satisfying assignment of a boolean formula $F(x_1, \dots, x_n)$. An assignment, A, of the variables $x_1, \dots, x_n$, is a feasible solution if it satisfies F. The cost of a feasible solution A is given by the binary number $A(x_1) \dots A(x_n)$. As a representative of polynomial valued problems we use MAXINDSET, the problem of computing the largest sized independent set in a given graph.

We use the following lemma to prove that approximate solutions for instances of LEXMAXSAT can not be computed with $n^{1-\epsilon}$ rounds of interaction, for all $\epsilon > 0$. Recall the language USAT, the set of boolean formulas, F, having a unique satisfying assignment, denoted by $s(F)$. It is well known that USAT is complete for NP under randomized reductions[VV86]. We showed in chapter 4 that USAT$\notin$ co-NP/*poly* unless the Polynomial Hierarchy collapses.

Lemma 6 *Assume* USAT$\notin$ co-NP/*poly. Let D be any deterministic polynomial time transducer and r any polynomial. Then for infinitely many n, there are boolean formulas $F_1, \dots, F_{r(n)}$, of length n, in* USAT, *such that for all j, $1 \leq j \leq r(n)$, D given $\{F_1, \dots, F_{r(n)}, s(F_1), \dots, s(F_{j-1}), s(F_{j+1}), \dots, s(F_{r(n)})\}$ can not compute $s(F_j)$.*

The existence of such "orthogonal" sets of formulas was used in [RCR91] to prove that the optimum solution can not be computed for LEXMAXSAT using less than $n^{1-\epsilon}$ rounds on inputs of length n. Using a novel construction, to build single difficult instance from many orthogonal formulae, and techniques from [RCR91] we can establish a lower bound the number of interactions needed to compute approximate solutions.

The following construction is used in the proof: Let $F_1, \ldots, F_{r(n)}$ be formulas of length n in USAT. Rename the variables so that only the variables $x_1, \ldots, x_m$ appear in each formula. Choose new variables $y_1, \ldots, y_s$, where $s = \lceil \log(r(n)) \rceil + 1$. Let $\vec{Y}$ denote the concatenation of these variables. For any integer k, let $\vec{k}$ denote the s-bit binary representation of k. Define the formula G as

$$[\ (\ \vec{Y} = \vec{1}\ ) \wedge F_1\ ]\ \bigvee\ \ldots\ \bigvee\ [\ (\ \vec{Y} = r(\vec{n})\ ) \wedge F_{r(n)}\ ]$$

The only satisfying assignments of G are assignments of $x_1, \ldots, x_m$ that satisfy some F_i with $\vec{Y} = \vec{i}$. The variables of G are ordered $y_1, \ldots, y_s, x_1, \ldots, x_m$. G is of length at most $2nr(n)$ and has exactly $r(n)$ satisfying assignments.

Theorem 18 *Let α be any constant. Unless* PH *collapses, α-approximate solutions can not be computed for instances of* LEXMAXSAT *in less than $n^{1-\epsilon}$ rounds, for any $\epsilon > 0$.*

PROOF. Fix an $\epsilon > 0$. Assume to the contrary that there is a protocol which computes α-approximate solutions for LEXMAXSAT using at most $n^{1-\epsilon}$ rounds on inputs of length n. Let D be the student in this protocol. Choose a polynomial $r(n)$ such that $(2nr(n))^{1-\epsilon} < \alpha\, r(n)$. Note that this is possible for any constant α. From D we construct a polynomial time machine D' which on input string $\{F_1, \ldots, F_{r(n)}, s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_{r(n)}\}$ works as follows:

- Checks that each s_i is a satisfying assignment of the corresponding F_i

- It simulates D on the input G which is constructed as described above from $F_1, \ldots, F_{r(n)}$. When D presents a solution A to the teacher, D' checks to see if the last m bits of A form a satisfying assignment of F_j. If so it stops and outputs this assignment. If the last m bits of A form a satisfying assignment of some other formula F_i, D' finds the least $t > i$ such that $s_t \in \{s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_{r(n)}\}$, constructs the corresponding assignment of G and resumes simulation assuming that the teacher provided this assignment. In all other cases D' outputs a default string and halts.

If PH does not collapse, by Lemma 1 for infinitely many n there are $r(n)$ formulas $F_1, \ldots, F_{r(n)}$, of length n, such that D' given the formulas and the satisfying assignments $s(F_1), \ldots, s(F_{j-1}), s(F_{j+1}), \ldots, s(F_{r(n)})$ can not compute $s(F_j)$ for any j. Consider the formula G constructed from $F_1, F_2, \ldots F_{r(n)}$ as described above. The formula G is of length $2nr(n)$ and has $r(n)$ satisfying assignments. Since D computes an α-approximate solution it must compute a satisfying assignment whose last m bits are the satisfying assignment of F_l where $l \geq \alpha\, r(n)$. Consider a teacher which provides satisfying assignments in lexicographic order. By assumption, D computes an α-approximate solution of G, against this teacher, with only $|G|^{1-\epsilon}$ *i.e.* $(2nr(n))^{1-\epsilon}$ rounds. By choice of $r(n)$ this is less than $\alpha\, r(n)$. Thus D must compute some satisfying assignment of G by itself. Assume that the first time this happens the last m bits give a satisfying assignment of F_i. By definition D' on input $F_1, F_2, \ldots F_{r(n)}, s(F_1), \ldots, s(F_{i-1}), s(F_{i+1}), \ldots, s(F_{r(n)})$ will compute the satisfying assignment of $s(F_i)$. This contradicts the choice of these orthogonal formulas. ∎

We can extend this result to show that unless PH collapses, α-approximate solutions for LEXMAXSAT can not be computed in $n^{1-\epsilon}$ rounds for any $\alpha \geq 1/p(n)$ where p is any polynomial.

The cost function of LEXMAXSAT can have exponentially many values of some instances. The technique used above only shows the existence of instances with a sublinear number of independent satisfying assignments. We are unable to extend the techniques used above to prove stronger lower bounds. In fact, in [KPS90] it is shown that a superpolynomial lower bound on the number of rounds to compute the optimum solution of LEXMAXSAT will show that the expressive powers of certain theories of bounded arithmetic are different. For problems such as MAXINDSET where the optimum value is linearly bounded we prove much tighter bounds. We state a general lemma which will be used to show lower bounds on the number of rounds required to compute approximate solutions for a variety of problems. Let FP/*poly* denote the set of functions that are computable by a deterministic polynomial time transducer with polynomial advice[4]. We say that an optimization problem $\mathcal{P}$ is α-approximable in FP/*poly* if there exists an α-approximation algorithm for $\mathcal{P}$ in FP/*poly*.

Lemma 7 *Let $\mathcal{P}$ be any optimization problem such that $\mathcal{P}$ is not α-approximable in* FP/*poly. Let D be any polynomial time transducer and r any polynomial. Then there are infinitely many n and instances $I_1, I_2, \ldots, I_{r(n)}$ of $\mathcal{P}$, of size n, such that D on input $\{I_1, I_2, \ldots, I_{r(n)}, h(I_1), \ldots, h(I_{j-1}), h(I_{j+1}), \ldots, h(I_{r(n)})\}$ can not compute an α-approximate solution for the instance I_j. Here $h(I)$ is a canonical optimal solution of the instance I.*

PROOF. Assume to the contrary there is a deterministic polynomial time transducer D and a polynomial $r(n)$ such that for all large n the above condition fails. This means that one of the following conditions holds

- D given $I_1, I_2, \ldots, I_{r(n)}$ and $h(I_2), \ldots, h(I_{r(n)})$ computes an α-approximate solution of I_1.
- D given the input $I_1, I_2, \ldots, I_{r(n)}$ and $h(I_1), h(I_3), \ldots, h(I_{r(n)})$ computes an α-approximate solution of I_2.

[4]A function f is in FP/*poly* if there is a function $\alpha : \mathcal{N} \rightarrow \Sigma^*$, such that $| \alpha(n) | \leq p(n)$ for some polynomial p, and a function g computable in polynomial time such that for all x, $f(x) = g(x, \alpha(| x |))$.

...

- D given $I_1, I_2, \ldots, I_{r(n)}$ and $h(I_1), \ldots, h(I_{r(n)-1})$ computes an α-approximate solution of $I_{r(n)}$.

Thus for any $r(n)$ tuple $(I_1, \ldots, I_{r(n)})$ of instances of length n there is some j for which D given the instances and $h(I_1), \ldots, h(I_{j-1}), h(I_{j+1}), \ldots, h(I_{r(n)})$.

The proof of Lemma 2 is an extension of techniques in[RCR91] and is omitted. We illustrate the lower bound technique for the optimization problem MAXINDSET. The following is a simple extension of the results of [FGL+91].

Fact 2 *Unless* NP *has quasipolynomial* $(2^{poly(\log n)})$ *sized circuits,* MAXINDSET *can not be* α*-approximated in* FP/*poly, where* $\alpha > 1/2^{\log^{1-\delta} n}$ *for any* $\delta > 0$.

Theorem 19 *Let* α *be a constant. Unless all languages in* NP *have quasipolynomial sized circuits,* α*-approximate solutions for* MAXINDSET *can not be computed with less than* $n^{1-\epsilon}$ *rounds of interaction for all* $\epsilon > 0$.

PROOF. For simplicity assume $\alpha = 1/t$ for some integer t. Assume to the contrary that there is a protocol to compute α-approximate solutions using less than $N^{1-\epsilon}$ on inputs of size N for some $\epsilon > 0$. Let D be the student in this protocol. Choose a polynomial r such that $2n(nr(n))^{1-\epsilon} < \alpha\, r(n)$, for large n. For instance let $r(n) = n^{\lceil 3/\epsilon \rceil + 1}$. From D we construct D' which on input $G_1, \ldots, G_{r(n)}, h(G_1), \ldots, h(G_{j-1}), h(G_{j+1}), \ldots, h(G_{r(n)})$, where each G_i is a graph of size n, works as follows:

1. Checks that each solution $h(G_i)$ is an independent set in the graph G_i.
2. Let $N = nr(n)$. D' simulates D on the graph G, the disjoint union of the graphs $G_1, \ldots, G_{r(n)}$ for upto $N^{1-\epsilon}$ rounds of interaction. Note that any solution for G is the union of independent sets of the component graphs. When D presents a solution $\mathcal{I}$, to the teacher, D' finds the smallest $k \neq j$ such that the component of $\mathcal{I}$ consisting of vertices in G_k contains less than $|h(G_k)|$

vertices. It then replaces this component of $\mathcal{I}$ with $h(G_k)$ and continues the simulation assuming that the teacher provided this solution. If no such k exists or if $N^{1-\epsilon}$ rounds have taken place, it outputs the largest independent set of G_j that it can identify from all rounds of interaction.

From Lemma 2 and Fact 1 we know that unless NP has quasipolynomial sized circuits, for infinitely many n there are graphs $G_1, \ldots, G_{r(n)}$, of size n, such that D' given the graphs and solutions $h(G_1), \ldots, h(G_{j-1}), h(G_{j+1}), \ldots, h(G_{r(n)})$, can not compute an α-approximate solution of G_j. Let G be the disjoint union of these graphs. It has size $N = nr(n)$ and its optimum solution is the union of the optimal solutions of each of the component graphs. Consider a teacher which when given a solution $\mathcal{I}$ finds the first graph G_i for which the component of $\mathcal{I}$ is not of size $|h(G_i)|$ and replaces this in $\mathcal{I}$ by $h(G_i)$. By assumption D works against this teacher in $N^{1-\epsilon}$ rounds. By choice of $r(n)$, if D computes an α-approximate solution of G it must compute an α-approximate solution of one the graphs for which the teacher does not provide an optimal solution. Suppose this not the case. Let $k = N^{1-\epsilon}$ and $i_1 \ldots, i_k$ be the sizes of the optimal independent sets provided by the teacher. It can be then shown that the solution computed by D has size at most $i_1 + i_2 + \ldots + i_k + (\alpha\, i_{k+1} - 1/t) + \ldots (\alpha\, i_{r(n)} - 1/t)$. Thus D computes a solution of size at most

$$
\begin{aligned}
& i_1 + \ldots + i_k + \alpha i_{k+1} + \ldots + \alpha i_{r(n)} - (r(n) - k)/t \\
< \; & i_1 + \ldots + i_k + \alpha(i_1 + \ldots + i_{r(n)}) - (r(n) - k)/t \\
\leq \; & nk - (r(n) - k)/t + \alpha\,(i_1 + \ldots + i_{r(n)}) \\
< \; & 2nk - r(n)/t + \alpha\,(i_1 + \ldots + i_{r(n)}) \\
< \; & \alpha\,(i_1 + \ldots + i_{r(n)})
\end{aligned}
$$

since by choice of r, $2nk < r(n)/t$. Thus if D computes an α-approximate solution of G then it must compute an α-approximate solution of one of the graphs by itself. Let the first graph for which it computes an α-approximate solution be G_j. Then D'

by definition, when given $G_1, \ldots, G_{r(n)}, h(G_1), \ldots, h(G_{j-1}), h(G_{j+1}), \ldots, h(G_{r(n)})$ computes an α-approximate solution of G_j. This contradicts the choice of the graphs. □

By carefully analyzing the technique used in the proof of this theorem we can prove the following:

Theorem 20 *Unless all languages in* NP *can be recognized by quasipolynomial sized circuits, there exist infinitely many instances of the problem* MAXINDSET *for which* $1/2^{\log^\alpha n}$*-approximate solutions can not be computed with* $n/2^{\log^\beta n}$ *rounds of interaction, where* $1 > \beta > \alpha > 0$.

Feige *et al* [FGL+91] show that MAXINDSET can not be approximated to within a factor $1/2^{\log^\alpha n}$ unless all languages in NP can be recognized by quasipolynomial time machines. The above theorem strengthens this by showing that MAXINDSET can not be approximated to the factor $1/2^{\log^\alpha n}$ even allowed $n/2^{\log^\beta n}$ rounds of interaction where $\beta > \alpha$. Since each interaction is guaranteed to result in a better solution, in $n/2^{\log^\beta n}$ rounds a trivial strategy results in a solution that is within a factor $1/2^{\log^\beta n}$ of optimal. It is surprising that *no* strategy can improve this slightly to get a solution that is within a factor f_1 of optimal. Thus the knowledge of an independent set which is within a factor $1/2^{\log^\beta n}$ is of no help in computing one which is within a factor $1/2^{\log^\alpha n}$ of the optimal.

5.3 An example where suboptimal solutions help

When studying optimization and approximation in the context of the counterexample model, a natural question that arises is: are there problems where a non-trivial strategy helps in reducing the number of rounds required to compute the desired answer?

In [KPS90] an artificial problem was presented where the trivial strategy requires two rounds to compute the optimum, whereas using a non-trivial strategy only one round is sufficient.

In this section we present an example of an optimization problem for which there is a simple protocol to compute a $2/3$-approximate solution with a sublinear number of counterexamples. Further we show that there is a protocol that computes α-approximate solutions using a sublinear number of rounds, for any $\alpha < 1$. Since following the trivial strategy requires linearly many rounds to do the same, this is an example where a non-trivial strategy is provably better than the trivial strategy.

To our knowledge, this is the first example of such a problem. This also gives some insight into how knowledge of some solutions may help in computing significantly better solutions.

Informally, the problem is: Given a number n, find a divisor d of n such that the size of d is as close to half the size of n as possible. Under the standard cryptographic assumption that factoring is hard, it is not possible to compute divisors in polynomial time. So, studying this problem in the counterexample model is basically addressing the question: how many divisors do we need to know before we know the complete factorization of a number? Formally we can define the maximization problem MIDDLE_DIVISOR as follows:

INSTANCE: An integer n of length N.

SOLUTIONS: Integers d such that $d|n$.

COST: $\sigma(n,d) = N/2 - |N/2 - |d||$, where $|d|$ denotes the size of d.

Fig 5.1 shows the cost function.

Note that, if we can compute a small divisor d of n we can also compute a large divisor of n, namely n/d. Hence, the real difficulty lies in computing a divisor whose size is close to half the size of the given number; hence the definition of σ, the cost function.

The optimal solution can clearly be computed in at most $N/2$ rounds using the

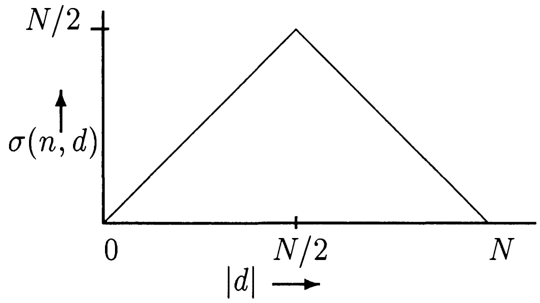

Figure 5.1: The cost function.

trivial strategy. On the other hand, the trivial strategy may require $\Omega(N)$ rounds in the worst case, even to compute an approximate solution. We present below a 2/3-approximating protocol for the above problem which uses at most $O(N/\log N)$ rounds.

INPUT: An integer n of length N.

PROTOCOL: The divisors of n are computed in stages. At each stage the divisor computed will be of size at least $\log N$ more than that of the divisor in the previous stage. Let s_k denote the solution computed at the end of stage k and t_k the counterexample provided by the teacher in stage k. Also since the cost of the solutions d and n/d are the same we assume that the teacher always provides a solution with size less than $\frac{N}{2}$. Any divisor whose size is between $\frac{N}{3}$ and $\frac{N}{2}$ is a 2/3-approximate solution.

1. Let $\mathcal{P}=\{p_i \mid i = 1, \ldots, m\}$ be the set of primes of size at most $\log N$. Let β_i be the highest power of p_i that divides n. Define $S = \Pi_{i=1}^{m} p_i^{\beta_i}$.

   STAGE 0: Compute $\mathcal{P}$ and S. If $|S| > \frac{N}{2}$ then divide S by primes in $\mathcal{P}$ till $|S| \leq \frac{N}{2}$. The initial solution is $s_0 = S$. Let t_0 be the counterexample provided if any.

2. STAGE $k \geq 1$: s_k is defined as follows:

$$s_k = \begin{cases} t_{k-1} & \text{if } |t_{k-1}| \geq \frac{N}{3} \\ \mathrm{lcm}(s_{k-1}, t_{k-1}) & \text{otherwise} \end{cases}$$

where $\mathrm{lcm}(a,b) = ab/\gcd(a,b)$. The solution s_k is presented to the teacher which either accepts or provides t_k as a counterexample.

The protocol halts when the student has computed a 2/3-approximate solution. The number of counterexamples is equal to the number of stages. The following propositions imply that the number of stages is at most $O(N/\log N)$.

Fact 3 *S divides s_k for all k.*

PROOF. If at the first stage $|S| \geq N/3$ it is accepted since it is a 2/3-approximate solution. Otherwise at stage k we set $s_k = lcm(s_{k-1}, t_{k-1})$. Hence, if S divides s_{k-1} then it clearly divides s_k. The proof follows by induction. □

Fact 4 *If g=$gcd(s_k, t_k)$ and $|t_k| > |s_k|$, then $|t_k| - |g| \geq \log N$.*

PROOF. Assume to the contrary that the size of g is more than $|t_k| - \log N$. Then t_k/g has size less than $\log N$ and is relatively prime to s_k/g. This implies that there is a prime p of size less than $\log N$ which divides t_k/g but not s_k/g. Thus there is a β such that p^β divides t_k but not s_k which contradicts since $S|s_k$ by Fact 2. □

Theorem 21 *For all k, either $\frac{N}{3} \geq |s_{k+1}| \geq |s_k| + \log N$ or the solution s_{k+1} is a 2/3-approximate solution.*

PROOF. Since the solution provided by the teacher, t_k, has cost greater than s_k and $|s_k| < \frac{N}{3}$ we have $|s_k| < |t_k|$. If $|t_k| \geq \frac{N}{3}$ then $s_{k+1} = t_k$, which by the assumption that the size of t_k is at most $\frac{N}{2}$, is a 2/3-approximate solution. Otherwise $s_{k+1} = \mathrm{lcm}(s_k, t_k)$ and size of s_{k+1} is $|s_k| + |t_k| - |g|$ where g is the gcd of s_k and t_k. Since s_k and t_k are both of size at most $\frac{N}{3}$, we have that $|s_{k+1}| \leq \frac{2N}{3}$. Also by Fact 3 $|s_{k+1}| > |s_k| + \log N$. If s_{k+1} is of size greater than $\frac{N}{3}$ then it is a 2/3-approximate solution. Otherwise its size is at most $\frac{N}{3}$. This establishes the theorem. □

The number of counterexamples required to compute a 2/3-approximate solution by this protocol is at most $N/\log N$. In fact, using a slightly involved and careful argument, we can show that for any $\alpha < 1$ there is an α-approximating protocol for MIDDLE_DIVISOR which runs in $O(N/\log N)$ rounds. The basic idea behind this protocol is that if s, the currently computed solution, is not α-approximate, then within c_α rounds the student can compute another solution whose size is at least $|s| + \log N$ if one exists. Here c_α is a constant that depends only on α. Thus the whole protocol needs at most $c_\alpha N/\log N$ rounds. The protocol computes the gcd's and lcm's of the divisors provided as counterexamples and takes their combinations to get a divisor with a better cost. The proof that the protocol really works relies on the following lemma.

Lemma 8 (Sunflower lemma) *For any $\alpha > 0$ let $c_\alpha = 1/(1-\alpha) + 1$. Let n be any integer and $\{d_1, \ldots d_{c_\alpha}\}$ be divisors of n such that sizes of d_i's are strictly increasing and none of the d_i's is an α-approximate solution. Then one can compute in polynomial time a divisor d of n with $|d| \geq |d_1| + \log N$.*

The reason why this is called the sunflower lemma is that the worst case arises when the divisors form a sunflower like structure with the common core representing the gcd of all the divisors.

The protocol is as follows:

INPUT: An integer n of length N.

PROTOCOL: Note that any divisor d whose size is between $\alpha N/2$ and $N/2$ is an α-approximate solution. Without loss of generality we can assume that α is less than $1 - 1/\log N$. If not we can use the following protocol till we obtain a solution of size $(1-1/\log N)N/2$ and then use the trivial strategy to obtain an α-approximate solution.

1. Let $\mathcal{P}=\{p_i \mid i = 1, \ldots, m\}$ be the set of small primes of size at most $\log N$. Let β_i be the highest power of p_i that divides n. Define $S = \Pi_{i=1}^{m} p_i^{\beta_i}$.

   STAGE 0: Compute $\mathcal{P}$ and S. If $|S| > \frac{N}{2}$ then divide S by primes in $\mathcal{P}$ till $|S| \leq N/2$. The initial solution is $s_0 = S$. Let t_0 be the counterexample provided if any. As a simplifying assumption we assume that all the solutions provided by the teacher contain all these small primes as factors.

2. STAGE $k \geq 1$: The algorithm works as follows: Given a set $\{d_1, d_2, \ldots, d_m\}$, of divisors of n, we construct from this a set R of relatively prime divisors of n with the following properties:

   - Each number in R divides some d_j, and has size at least $\log N$.
   - At most one element of R divides all the numbers.
   - If an element r does not divide d_j then r and d_j are relatively prime.

   If any element r in R does not divide, and is hence relatively prime with, d_j then rd_j is a new divisor of n. Further if the size of r is at most $(1-\alpha)N$ then the new divisor has size at most $\alpha N/2 + (1-\alpha)N$ and since $|r|$ is at least $\log N$, this gives us a solution which is of size $\log N$ greater than the previous solution. If no such r exists, R is a set of relatively prime divisors of n each of size at least $(1-\alpha)N$. But then there can be no more than $1/(1-\alpha)$ such numbers. We show later that from $\{d_1, \ldots, d_m\}$ we can construct R of size

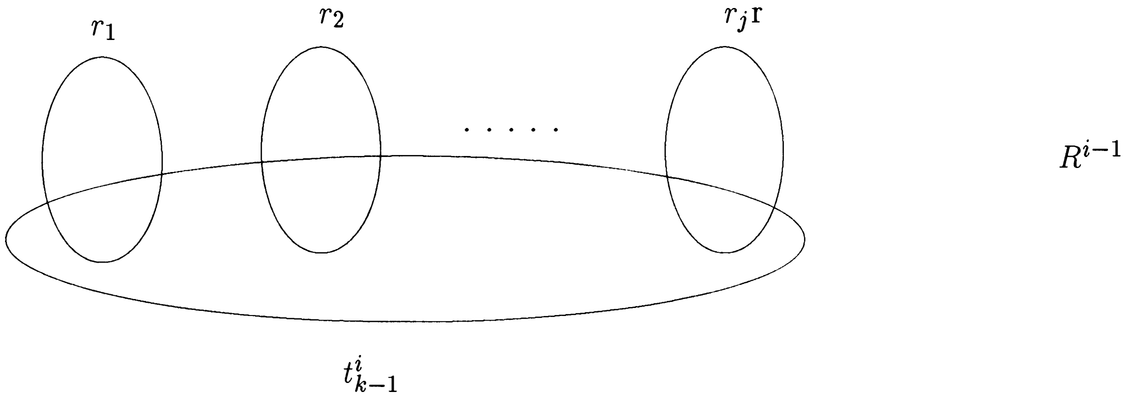

Figure 5.2: R^i is the set consisting of the distinct regions . Intersection of two regions is the gcd of the corresponding numbers.

at least m. Hence we need to consider only $1/(1-\alpha)+2$ solutions, before we get an element of R of size less than $(1-\alpha)N$.

Let $s^0_{k-1} = s_{k-1}$ and $R^0 = \{s_{k-1}\}$. Clearly R^0 satisfies the required properties.

Repeat the following until a divisor of size $|s_{k-1}| + \log N$ is found.

a. Present s^{i-1}_{k-1} to the teacher.

b. Let t^i_{k-1} be the response of the teacher (if any).

c. From R^{i-1} we construct the new set of integers as follows

$$R^i_1 = \{x \mid x = gcd(r, t^i_{k-1}), \text{ if } x \neq 1, r \in R^{i-1}\}$$

$$R^i_2 = \{x \mid x = r/gcd(r, t^i_{k-1}), \text{ if } x \neq 1 r \in R^{i-1}\}$$

R^i= $R^i_1 \cup R^i_2 \cup \{\Delta\}$ where $\Delta = t^i_{k-1}/\Pi_{r \in R^{i-1}} gcd(r, t^i_{k-1})$.

It can be verified that if R^{i-1} satisfies the properties enumerated above then so does R^i. Figure 5.2 pictorially describes the construction. Think of each number as the (multi)set of primes that are its factors. Then the set R^i represents all non empty regions of the Venn diagram for the numbers. Each region other than the intersection of all the sets is of size at least $\log N$ because, by assumption, each solution provided by the teacher has all the small prime powers as factors. Thus all small prime powers are in the intersection of all sets and any other non empty region has only large primes and is hence of size at least $\log N$.

d. Compute the divisor with the largest cost that can be obtained as a product of numbers in R^i. Let s^i_{k-1} be the solution thus found.

The following lemmas establishes a bound on the number of iterations in each stage.

Lemma 9 *At stage k, $|R^i_k| \geq i$.*

PROOF. This is proved by induction on i. This is clearly true for $i = 1$ since the only element is s^0_{k-1}. By definition of the protocol the Student finds the best possible solution that can be obtained by multiplying numbers in R^{i-1}. The best such solution, s^i_{k-1} is presented to the teacher. The new solution, t^i_k, that is provided by the teacher must have a cost better than that of s^i_{k-1}. By definition of R^i each number of R^{i-1} contributes to at least one unique element of R^i, $|R^i_1 \cup R^i_2| \geq |R^{i-1}|$. Also one of the following cases must occur.

1. The new solutions is relatively prime to all the numbers in R^{i-1}. In this case $R^i = R^{i-1} \cup \{t^i_k\}$ and hence has one more element.

2. There is a number $r \in R^{i-1}$ such that both $\gcd(r, t^i_k)$ and $r/\gcd(r, t^i_k)$ are greater than 1. In this case $R^i_1 \cup R^i_2$ has at least one element more than R^{i-1}.

3. For no r are both $\gcd(r, t^i_k)$ and $r/\gcd(r, t^i_k)$ greater than 1. Note that this includes the first case. In this case the number Δ is not equal to 1. For otherwise t^i_k would just be a product of numbers from R^{i-1}. Thus this provides a new element in R^i.

Hence for all i, R^i has at least i elements. □

Lemma 10 *At each stage k the number of iterations is at most $c_\alpha = 1/(1-\alpha)+1$*

PROOF. By definition of R^i, $\gcd(r_1, r_2)$ =1 for all $r_1, r_2 \in R^i$. If $r \in R^i$ and r does not divide all the numbers in $\{s^0_{k-1}, \ldots, s^{i-1}_{k-1}\}$ then there exists a divisor s^j_{k-1} such that rs^j_{k-1} is a new divisor. Hence if $|r| < (1-\alpha)N$ the size of the new solution is less than $N/2 + (1-\alpha)N/2$ as $|s^j_{k-1}| < \alpha N/2$. Since the size of r is at least $\log N$ this gives us the required improvement. If this is not the case each $r \in R^i$, except possibly the one element that divides all the numbers, is of size at least $(1-\alpha)N$. By the above lemma there are at least $i-1$ such elements. Since each such r divides n and the gcd of two such elements is 1, the product of all these regions, which is of size at least $(i-1)(1-\alpha)N$, divides n. Thus if $i > c_\alpha$ there must be an element of size less than $(1-\alpha)N$. □

The existence of a $o(N)$ non-trivial protocol for MIDDLE_DIVISOR should not be surprising. In fact, it reinforces the belief that factorization is easier than the canonical NP-hard problems since none of these problems are known to be solvable using $o(N)$ rounds. We conjecture that using more sophisticated number theoretic techniques it is possible to devise a more efficient protocol for MIDDLE_DIVISOR.

5.4 Protocols with Guarantees

Until now, we have considered protocols in which the teacher T could provide any better solution as a counterexample. Since there was no guarantee on the counterexamples, the worst-case teacher could force the student to take many rounds by providing least helpful and marginally better counterexamples. In this section we impose restrictions on T by insisting that she provide significantly better counterexamples, i.e., counterexamples whose cost is better than the cost of the student's solution by at least $f(n)$ for some function f. This additive guarantee is intended to model heuristics which improve the solution by at least $f(n)$ whenever they succeed. To take care of boundary conditions, we allow T to present the optimal solution in the last round even though it may not be better than the student's solution by the required amount.

As usual, we shall measure the complexity of computing the optimal or approximate solutions to NPO problems in this model in terms of the number of counterexamples required by the student given the least cooperative teacher. Our results show that surprisingly, the student is unable to benefit from the substantially improved solutions provided by the teacher and cannot do much better than the trivial strategy.

Consider the NPO problem MAXCLIQUE. If there is an additive guarantee of $f(n)$ on the counterexamples provided by the teacher, then by using the trivial strategy the student can compute the optimal solution using $n/f(n)$ rounds of interaction. The following theorem shows that the student cannot do much better.

Theorem 22 *Unless* PH *collapses, no student can solve* MAXCLIQUE *using less than $n^{1-\varepsilon}/f(n)$ counterexamples for any $\varepsilon > 0$, given an arbitrary teacher with an $f(n)$ additive guarantee on counterexamples.*

Similarly, for any constant α, by following the trivial strategy a student can α-approximate MAXCLIQUE using at most $\alpha * n/f(n)$ rounds. The following theorem shows that this strategy is almost optimal.

Theorem 23 *Unless* NP *has quasipolynomial sized circuits, no student can compute α-approximate solutions of* MAXCLIQUE *using less than $n^{1-\varepsilon}/f(n)$ counterexamples for any $\varepsilon > 0$, given an arbitrary teacher with an $f(n)$ additive guarantee on counterexamples.*

It is possible to extend these techniques to prove slightly stronger results. One can also prove lower bounds on the number of counterexamples required to compute LEXMAXSAT in this model. However, as is the case with the earlier model, these bounds are not tight. This is because LEXMAXSAT can have exponentially many feasible solutions with different costs, whereas our techniques (see Theorem 18) are limited to formulas having sublinear number of widely spaced satisfying assignments. For polynomially bounded $f(n)$, we can modify the proof of Theorem 18 to obtain a lower bound of $n^{1-\varepsilon}$. For large $f(n)$, we can get a lower bound of $n^{1-\varepsilon}/f(n)$.

Bibliography

[BG81] C.H. Bennett and J. Gill. Relative to a random oracle, $P^A \neq NP^A \neq coNP^A$. *SIAM Journal on Computing*, 10(1):96–112, 1981.

[Blu91] A. Blum. *Algorithms for Approximate Graph Coloring.* Ph.D. dissertation, M.I.T., 1991.

[Bus86] S. Buss. *Bounded Arithmetic.* Studies in Proof Theory 3. Bibliopolis, Naples, 1986.

[CK73] C.C. Chang and H.J. Keisler. *Model Theory.* North-Holland, Amsterdam, 1973.

[Coo71] S.A. Cook. The complexity of theorem proving procedures. In *3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.

[CP89] P. Crescenzi and A. Panconesi. Completeness in approximation classes. In *Lectures Notes in Computer Science # 380*, pages 116–126. Springer-Verlag, 1989. Proceedings of the FCT.

[Edm65] J. Edmonds. Minimum partition of matroids into independent sets. *Journal of Research of the National Bureau of Standards(B)*, 69:67–72, 1965.

[Fag74] R. Fagin. Generalized first-order spectra, and polynomial-time recognizable sets. In R.M. Karp, editor, *Complexity and Computations.* AMS, 1974.

[FGL+91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *32nd Symposium on Foundation of Computer Science*, pages 2–12, 1991.

[GJ76] M.R. Garey and D. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23:43–49, 1976.

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

[HS65] J. Hartmanis and R.E. Stearns. On the computational complexity of algorithms. *Transactions of the AMS*, 117:285–306, 1965.

[HS87] D. Hochbaum and D. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34, 1987.

[Imm80] N. Immerman. *First Order Expressibility as a New Complexity Measure.* Ph.D. dissertation, Cornell University, Ithaca, New York, August 1980.

[Imm88] N. Immerman. Nondeterministic space is closed under complement. In *Proceedings of Structure in Complexity Theory Third Annual Conference*, pages 112–115. IEEE Computer Society Press, 1988.

[Joh74] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[JPY88] D. Johnson, C. Papadimitriou, and M. Yannakakis. How easy is local search. *Journal of Computer and System Sciences*, 37:179–200, 1988.

[Kan92] V. Kann. *On the Approximability of NP-complete Optimization Problems.* Ph.D. dissertation, Royal Institute of Technology, Stockholm, Sweden, May 1992.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Kar86] R.M. Karp. Combinatorics, complexity and randomness. *Communications of the ACM*, 29(2):98–109, February 1986. Turing Award Lecture.

[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.

[KT90] P.G. Kolaitis and M.N. Thakur. Logical definability of NP-optimization problems. Computer and Information Sciences Technical Report UCSC-CRL-90-48, University of California, Santa Cruz, 1990.

[KT91] P.G. Kolaitis and M.N. Thakur. Approximation properties of NP-minimization classes. In *Structure in Complexity Theory, Sixth Annual Conference.* IEEE Computer Society Press, 1991.

[Lin65] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

[LK73] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 11:972–989, 1973.

[OM87] P. Orponen and H. Mannila. On approximation preserving reductions: complete problems and robust measures. Technical report, University of Helsinki, 1987.

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

[PM81] A. Paz and S. Moran. NP-optimization problems and their approximation. *Theoretical Computer Science*, 15:251–277, 1981.

[PR90] A. Panconesi and D. Ranjan. Quantifiers and approximation. In *22nd Annual ACM Symposium on Theory of Computing*, pages 446–456. ACM, 1990.

[PS82] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[PY88] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. In *20^{th} ACM Symposium on Theory of Computing*, pages 229–234, 1988.

[RCR91] D. Ranjan, S. Chari, and P. Rohtagi. Improving known solutions is hard. In *Proceedings of the* 18^{th} *ICALP*, pages 381–392. Springer-Verlag, 1991. Lecture Notes in Computer Science # 510.

[Sha49] C.E. Shannon. The synthesis of two terminal switching circuits. *Bell Systems Technical Journal*, 28(1):59–98, 1949.

[Sip92] M. Sipser. The history and status of the P versus NP question. In *24th Annual ACM Symposium on Theory of Computing*, pages 603–618. ACM, 1992. Invited Lecture.

[Sze87] R. Szelepcsényi. The method of forcing for nondeterministic automata. *The Bulletin of the European Association for Theoretical Computer Science*, 33:96–100, October 1987.

[vN53] J. von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, 2:5–12, 1953.

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

[Yab59a] S. Yablonski. The algorithmic difficulties of synthesizing minimal switching circuits. In *Problemy Kibernetiki 2*, pages 75–121. Moscow, Fizmatgiz, 1959. Translation in *Problems of Cybernetics*, Pergamon Press, 401-457, 1961.

[Yab59b] S. Yablonski. On the impossibility of eliminating brute force search in solving some problems of circuit theory. *Doklady AN SSSR*, 124:44–47, 1959.

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