RC 20745 (02/26/97)
Computer Science/Mathematics

IBM Research Report

A complete classification of maximization problems derived from Boolean constraint satisfaction

Sanjeev Khanna

Fundamental Mathematics Research Department
Bell Laboratories
700 Mountain Avenue
NJ 07974

Madhu Sudan, David P. Williamson

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

A Complete Classification of the Approximability of Maximization Problems Derived from Boolean Constraint Satisfaction*

Sanjeev Khanna† Madhu Sudan‡ David P. Williamson§

March 6, 1997

Abstract

In this paper we study the approximability of boolean constraint satisfaction problems. A problem in this class consists of some collection of "constraints" (i.e., functions $f : \{0,1\}^k \rightarrow \{0,1\}$); an instance of a problem is a set of constraints applied to specified subsets of n boolean variables. Schaefer earlier studied the question of whether one could find in polynomial time a setting of the variables satisfying all constraints; he showed that every such problem is either in P or is NP-complete. We consider optimization variants of these problems in which one either tries to maximize the number of satisfied constraints (as in MAX 3SAT or MAX CUT) or tries to find an assignment satisfying all constraints which maximizes the number of variables set to 1 (as in MAX CUT or MAX CLIQUE). We completely classify the approximability of all such problems. In the first case, we show that any such optimization problem is either in P or is MAX SNP-hard. In the second case, we show that such problems fall precisely into one of five classes: solvable in polynomial-time, approximable to within constant factors in polynomial time (but no better), approximable to within polynomial factors in polynomial time (but no better), not approximable to within any factor but decidable in polynomial time, and not decidable in polynomial time (unless P = NP). This result proves formally for this class of problems two results which to this point have only been empirical observations; namely, that NP-hard problems in MAX SNP always turn out to be MAX SNP-hard, and that there seem to be no natural maximization problems approximable to within polylogarithmic factors but no better.

Keywords : Approximation algorithms, combinatorial optimization, complete problems, computational complexity, computational classes, constraint satisfaction problems, hardness of approximation, polynomial reductions.

AMS Subject Classification : 68Q25.

---

*An early version of this paper appeared as a ECCC Tech. Report [10]. This version extends and subsumes the earlier version.

†`sanjeev@theory.stanford.edu`. Department of Computer Science, Stanford University, Stanford, CA 94305.
Supported by a Schlumberger Foundation Fellowship, an OTL grant, and NSF Grant CCR-9357849.

‡`madhu@watson.ibm.com`. IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.

§`dpw@watson.ibm.com`. IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.

1 Introduction

In this paper, we study the approximability of optimization versions of boolean constraint satisfaction problems (CSPs). A boolean CSP consists of a collection $\mathcal{F}$ of boolean functions $f : \{0,1\}^k \rightarrow \{0,1\}$ called *constraints*. An instance of such a problem is a set of "constraint applications". Each application is a constraint drawn from $\mathcal{F}$ and applied to a specified subset of n boolean variables.

The decision version of a boolean constraint satisfaction problem asks whether there is an assignment to the variables such that all constraint applications are satisfied (that is, for each application the specified boolean function evaluates to 1 on the given subset of variables). For a collection of constraints $\mathcal{F}$, we call this problem SAT($\mathcal{F}$). Thus, for example, 3SAT is a decision version of boolean CSP with the constraint functions $f_1(x,y,z) = x \vee y \vee z$, $f_2(x,y,z) = \bar{x} \vee y \vee z$, and so on. Schaefer [15] studied the decision version of these problems and proved a remarkable result: for every such problem, either it is in P or it is NP-complete. This dichotomy is especially interesting in light of Ladner's theorem [12], which states that if P$\neq$NP, then there exist infinitely many problems of complexity between P and NP-complete. Thus although problems SAT($\mathcal{F}$) could in principle display a wide range of complexity, they in fact fall into distinct and quite separate classes (assuming P$\neq$NP). An additional property of Schaefer's result is that his characterization of the problems in P is compact. He gives six classes of functions, and if all functions in $\mathcal{F}$ fall entirely within any one of these classes, then SAT($\mathcal{F}$) is in P, otherwise it is NP-complete.

In this paper, we consider two different maximization versions of SAT($\mathcal{F}$) and completely classify the approximability of all such problems. In so doing, we find that these optimization problems also fall into distinct and separate classes, bypassing the many intermediate levels of approximability which are possible in principle. As we describe later, this classification proves formally for these problems some results which to this point have only been empirical observations. Furthermore, our classification of the problems also has a compact description. For both types of maximization versions of SAT($\mathcal{F}$), we refine Schaefer's classes, and the level of approximability of a problem for a given $\mathcal{F}$ is determined by which of these classes contain $\mathcal{F}$.

In the first maximization version of SAT($\mathcal{F}$) that we consider, for each instance of a problem we are also given a nonnegative weight w_i for each constraint application i, and we must try to find an assignment to the variables which maximizes the weight of the satisfied constraint applications. For any set of constraints $\mathcal{F}$, we call this associated maximization problem MAX CSP($\mathcal{F}$), and we call the class of all such problems MAX CSP. It follows almost immediately from its definition that MAX CSP is contained in the well-studied class MAX SNP. Conversely, it also contains many of its complete problems. For example, MAX 3SAT and MAX CUT can be cast as MAX CSP($\mathcal{F}$) problems. We show that each problem MAX CSP($\mathcal{F}$) is either solvable exactly in polynomial time or is MAX SNP-hard. Thus there is no problem in this class which has an approximation scheme but is not solvable in polynomial time. This result has been obtained independently by Creignou [4]; however our result is stronger in certain technical senses which we discuss later.

In the second maximization version of SAT($\mathcal{F}$) that we consider, for each instance of the problem we are also given a nonnegative weight w_i for each boolean variable, and we must try to find a boolean assignment of maximum weight that satisfies all constraint applications. For any set of constraints $\mathcal{F}$, we call this associated maximization problem MAX ONE($\mathcal{F}$); for example, MAX CUT and MAX CLIQUE can be cast as MAX ONE($\mathcal{F}$) problems. We show that each problem MAX ONE($\mathcal{F}$) must fall into one of five classes: first, it is solvable exactly in polynomial time; second, it can be approximated to within some constant factor but no better; third, it can be approximated to within some factor that is polynomial in the number of variables, but no better; fourth, it is NP-complete to find a satisfying assignment of non-zero value; fifth, it is NP-complete to find any satisfying assignment.

The central idea of our proofs is a new concept we call an *implementation*. Given a set of constraints $\mathcal{F}$, we show that if $\mathcal{F}$ has certain properties, then it can be used to enforce other constraint functions f. We show that under suitable conditions, implementations can be composed, so that the constraints of $\mathcal{F}$ can be used to implement the constraints of other problems (such as MAX CUT or MAX CLIQUE) whose approximability is well known. The central difficulty of the proofs is showing that this can be done in an exhaustive way for all possible sets of constraints $\mathcal{F}$. Our definition of an implementation here is inspired by the notion of a gadget in Bellare et al. [3] and we unify their many definitions (they have different definitions for every f and $\mathcal{F}$ that they consider) into a single one. Our definition has in turn been used by Trevisan et al. [16] to derive improved hardness of approximation results and improved approximation algorithms.

Our results prove formally for these classes of problems some results about approximability which to this point have only been empirical observations. For example, the study of MAX SNP has revealed so far that every NP-hard MAX SNP problem is also hard to approximate to within some constant factor. Our result on MAX CSP serves as a formal basis for this empirical observation. Similarly, in the search for polynomial-time approximation algorithms, optimization problems so far either have exact algorithms, or approximation schemes, or constant or (poly)logarithmic or polynomial approximation algorithms – but this list is virtually exhaustive. There have been no "natural" problems that are approximable to within intermediate factors, such as $2^{\log^\epsilon n}$ or $\log\log n$ and no better. In addition, for many natural optimization problems the best known approximation algorithm guarantees logarithmic factor approximability, and yet none of them is a *maximization* problem. Once again, our results show that these observations are not simply due to a lack of knowledge, but have some formal basis.

One of the original motivations for this work was to find some simple rules which characterize the approximability of any given optimization problem. However the very general question, "Given an optimization problem, determine its approximability" is undecidable (by Rice's Theorem). Hence we turned to restricted classes of uniformly presented optimization problems and this allowed us to achieve our goal. A natural next step in this research agenda is to broaden the classes of problems covered by this approach. Khanna et al. [11] have already extended this line of research to minimization problems, obtaining a complete classification for MIN CSP and MIN ONES. Other possible research directions include: (1) extending the function families that are studied (to include, say, functions of bounded range or functions of unbounded arity); (2) placing restrictions on the nature of the interaction between constraints and variables (such as bounding the number of times a variable can appear in a constraint). One such restriction which has been explored by Khanna and Motwani [8] is the case where the interaction graph of the constraint applications and the variables is planar.

The rest of the paper is structured as follows. In Section 2, we present some definitions and state our main results. We also state the ways in which our result for MAX CSP($\mathcal{F}$) strengthens that of Creignou. Section 3 defines implementations and states some basic properties of implementations. Section 4 presents the proof of the MAX CSP($\mathcal{F}$) result. Finally, Section 5 gives the proof of our result for MAX ONE($\mathcal{F}$).

2 Definitions and Main Results

We begin with some definitions. A constraint f is as defined above, and a constraint application is a pair $(f, (i_1, \ldots, i_k))$, where the $i_j \in [n]$ indicate to which k of the n boolean variables the constraint is applied. We require that $i_j \neq i_{j'}$ for $j \neq j'$. While the distinction between constraints and constraint applications is important, we will often blur this distinction in the rest of this paper. In particular we may often let the constraint application $C = (f, (i_1, \ldots, i_k))$ refer just to the constraint f. In particular, we will often use the expression "$C \in \mathcal{F}$" when we mean "$f \in \mathcal{F}$, where

f is the first part of C".

For a given set of constraints $\mathcal{F}$, SAT$(\mathcal{F})$, MAX CSP$(\mathcal{F})$, and MAX ONE$(\mathcal{F})$ are as defined above. If for a given instance of MAX CSP$(\mathcal{F})$ (MAX ONE$(\mathcal{F})$) the weights $w_i = 1$ for all i, we call this an *unweighted* instance of MAX CSP$(\mathcal{F})$ (MAX ONE$(\mathcal{F})$). An instance which is not unweighted is *weighted*. We define the class of problems MAX CSP (MAX ONE) to be the set of all problems in MAX CSP$(\mathcal{F})$ (MAX ONE$(\mathcal{F})$) taken over all possible sets of constraints $\mathcal{F}$.

We now need some definitions from the theory of approximation algorithms. Given an NPO (NP Optimization) problem Π and a function $\alpha : \mathcal{Z}^+ \to \mathcal{Z}^+$ (with $\alpha(\cdot) \geq 1$), we say that an algorithm A is an *α-approximation algorithm* for Π if for every instance $\mathcal{I}$ of Π of size n, A produces, in time polynomial in n, a solution s to $\mathcal{I}$ of value in the range $[\mathrm{OPT}(\mathcal{I})/\alpha(n), \alpha(n)\mathrm{OPT}(\mathcal{I})]$. We say Π is *α-approximable* if such an algorithm exists. We define APX to be the class of all NPO problems which have constant-factor approximation algorithms, and poly-APX to be the class of NPO problems which have polynomial-factor approximation algorithms.

We also need to define what it means to be hard to approximate a problem Π to within a factor of α. For a function $\alpha : \mathcal{Z}^+ \to \mathcal{Z}^+$ with $\alpha(\cdot) \geq 1$, an NP maximization problem Π is hard to approximate to within a factor of α if there exists a polynomial time reduction f from SAT to Π which maps instances of SAT of length n to instances of Π of length $l(n)$ and for every n and for any two instances ϕ_1, ϕ_2 of size n of SAT such that $\phi_1 \in$ SAT and $\phi_2 \notin$ SAT, $\mathrm{OPT}(f(\phi_1))/\mathrm{OPT}(f(\phi_2)) \geq \alpha(l(n))$. Thus a problem Π is APX-hard if there exists a constant function $\alpha_\Pi > 1$ such that Π is hard to approximate to within α_Π. A problem Π is poly-APX-hard if there exists an $\epsilon > 0$ such that Π is hard to approximate to within n^ϵ. A problem is APX-complete (poly-APX-complete) if it is in APX (poly-APX) and is APX-hard (poly-APX-hard).

It is usual to define completeness for approximation classes in terms of reducibility, rather than the hardness of approximation of the problem. However, Khanna et al. [9] have shown that these two notions are equivalent provided the right approximation preserving reductions are used. We will not go into these definitions here, and refer the reader to their paper for details.

We now describe the main constraint classes that are identified by Schaefer's and our results. We say a constraint f is *0-valid* (*1-valid*) if $f(\bar{0}) = 1$ ($f(\bar{1}) = 1$). It is *weakly positive* (*weakly negative)* if it can be expressed in conjunctive normal form, with all the disjuncts having at most one negated literal (positive literal). A constraint f is *affine* if it can be expressed as a conjunction of linear equalities over GF(2). And, finally, a constraint f is *2CNF* if it can be expressed in conjunctive normal form with all disjuncts having at most two literals.

The above six constraint classes can now be used to describe Schaefer's result. In what follows we use phrases such as "$\mathcal{F}$ is 0-valid" to imply that "every function $f \in \mathcal{F}$ is 0-valid". We stress that when we say something like "$\mathcal{F}$ is 0-valid or 1-valid", we mean that "every function in $\mathcal{F}$ is 0-valid or every function in $\mathcal{F}$ is 1-valid".

Theorem 2.1 [Schaefer's Theorem [15]] For any constraint set $\mathcal{F}$, SAT$(\mathcal{F})$ is either in P or is NP-complete. Furthermore, SAT$(\mathcal{F})$ is in P if and only if $\mathcal{F}$ is 0-valid or 1-valid or weakly positive or weakly negative or affine or 2CNF.

We now provide the definitions required to state our main classification result for MAX CSP. For starters, observe that for the approximability of MAX CSP$(\mathcal{F})$ does not change by removing (or adding) functions from $\mathcal{F}$ which are not satisfiable. Hence, given a constraint set $\mathcal{F}$, we define the constraint set $\mathcal{F}'$ to be the set of constraints f in $\mathcal{F}$ which are satisfiable. We also need to define one more class of constraints before we can give our result: we say a constraint f of arity k is *2-monotone* if there exist indices $i_1, \ldots, i_p \subset \{1, \ldots, k\}$ and $j_1, \ldots, j_q \subset \{1, \ldots, k\}$ such that $f(X_1, \ldots, X_k) = (X_{i_1} \wedge \cdots \wedge X_{i_p}) \vee (\bar{X}_{j_1} \wedge \cdots \wedge \bar{X}_{j_q})$.

Theorem 2.2 (MAX CSP **Classification Theorem**) For every constraint set $\mathcal{F}$, the problem MAX CSP($\mathcal{F}$) is always either in P or is APX-complete. Furthermore, it is in P if and only if $\mathcal{F}'$ is 0-valid or 1-valid or 2-monotone.

We need to give two more classes of constraints to give our result for MAX ONE. We say a constraint is *affine with width* 2 if it can be expressed as a conjunction of linear equalities over GF(2) with at most two variables per equality constraint. A constraint f is *strongly 0-valid* if it is satisfied by any assignment with less than or equal to 1 ones.

Theorem 2.3 (MAX ONE **Classification Theorem**) For every constraint set $\mathcal{F}$, MAX ONE($\mathcal{F}$) is either solvable exactly in P or APX-complete or poly-APX-complete or decidable but not approximable to within any factor or not decidable. Furthermore,

1. If $\mathcal{F}$ is 1-valid or weakly positive or affine with width 2, then MAX ONE($\mathcal{F}$) is in P.
2. Else if $\mathcal{F}$ is affine then MAX ONE($\mathcal{F}$) is APX-complete.
3. Else if $\mathcal{F}$ is strongly 0-valid or weakly negative or 2CNF then MAX ONE($\mathcal{F}$) is poly-APX complete.
4. Else if $\mathcal{F}$ is 0-valid then SAT($\mathcal{F}$) is in P but finding a solution of positive value is NP-hard.
5. Else finding any feasible solution to MAX ONE($\mathcal{F}$) is NP-hard.

As stated previously, Theorem 2.2 was independently discovered by Creignou [4]. One fundamental point of difference between our result and hers is that we do not allow the use of variable replication in a constraint application. This is enforced by our definition of constraint application which insists that the indices $i_1, \ldots, i_k$ must be distinct. Our theorem shows that this does not ultimately matter, but this is not obvious a priori. For instance, a problem whose approximability has often been studied is the MAX EXACT kSAT problem: Given a collection of clauses of length exactly k, satisfy as many as possible. This problem is known to be approximable to within $1 + 2^{-k}/(1 - 2^{-k})$. However this problem cannot be captured as a MAX CSP problem under Creignou's notion of constraint application. The related problem that she can capture is MAX kSAT: Given a collection of clauses of length *at most* k, satisfy as many as possible. The best known approximation for this problem is only slightly smaller than 4/3 [17, 6, 7, 2]. Thus replications do end up altering the approximability of optimization problems and we take care to study the approximability of problems without the use of replications.

3 Implementations

We now describe the main technique used in this paper to obtain hardness of approximation results. Suppose we want to show that for some constraint set $\mathcal{F}$, the problem MAX CSP($\mathcal{F}$) is APX-hard. We will start with a problem that is known to be APX-hard, such as MAX CUT, which is the same as MAX CSP($\{X \oplus Y\}$). We will then have to reduce this problem to MAX CSP($\mathcal{F}$). The main technique we use to do this is to "implement" the constraint $X \oplus Y$ using constraints from the constraint set $\mathcal{F}$. We show how to formalize this notion next and then show how this translates to approximation preserving reductions.

Definition 3.1 [Implementation] A collection of constraint applications $C_1, \ldots, C_m$ over a set of variables $\vec{X} = \{X_1, X_2, ..., X_p\}$ and $\vec{Y} = \{Y_1, Y_2, ..., Y_q\}$ is called an α-implementation of a constraint $f(\vec{X})$ for a positive integer α iff the following conditions are satisfied:

(a) no assignment of values to $\vec{X}$ and $\vec{Y}$ can satisfy more than α constraints from $C_1, \ldots, C_m$.

(b) for any assignment of values to $\vec{X}$ such that $f(\vec{X})$ is true, there exists an assignment of values to $\vec{Y}$ such that precisely α constraints are satisfied,

(c) for any assignment of values to $\vec{X}$ such that $f(\vec{X})$ is false, no assignment of values to $\vec{Y}$ can satisfy more than $(\alpha - 1)$ constraints.

An implementation which satisfies the following additional property is called a *strict* α-implementation.

(d) for any assignment to $\vec{X}$ which does not satisfy f, there always exists an assignment to $\vec{Y}$ such that precisely $(\alpha - 1)$ constraints are satisfied.

A collection of m constraints is a *perfect* implementation of f if it is an m-implementation of f. A constraint set $\mathcal{F}$ (strictly / perfectly) implements a constraint f if there exists a (strict / perfect) α-implementation of f using constraints of $\mathcal{F}$ for some $\alpha < \infty$. We refer to the set $\vec{X}$ as the *constraint variables* and the set $\vec{Y}$ as the *auxiliary variables.*

A constraint f 1-implements itself strictly and perfectly. While properties (a)-(c) have perhaps been used implicitly elsewhere, property (d) is more strict (hence the name), but turns out to be critical in composing implementations together. The following lemma shows that the implementations of constraints compose together, if they are strict or perfect.

Lemma 3.2 If $\mathcal{F}_f$ strictly (perfectly) implements a constraint f, and $\mathcal{F}_g$ strictly (perfectly) implements a constraint $g \in \mathcal{F}_f$, then $(\mathcal{F}_f \setminus \{g\}) \cup \mathcal{F}_g$ strictly (perfectly) implements the constraint f.

Proof: Let $C_1, \ldots, C_{m_1}$ be constraint applications from $\mathcal{F}_f$ on variables $\vec{X}, \vec{Y}$ giving an α_1-implementation of f with $\vec{X}$ being the constraint variables. Let $C'_1, \ldots, C'_{m_2}$ be constraint applications from $\mathcal{F}_g$ on variable set $\vec{X}', \vec{Z}'$ yielding an α_2-implementation of g. Further let the first β constraints of $C_1, \ldots, C_{m_1}$ be applications of the constraints g.

We create a collection of $m_1 + \beta(m_2 - 1)$ constraints from $(\{\mathcal{F}_f \setminus \{g\}) \cup \mathcal{F}_g$ on a set of variables $\vec{X}, \vec{Y}, \vec{Z}'_1, \ldots, \vec{Z}'_\beta$ as follows: We include the constraint applications $C_{\beta+1}, \ldots, C_{m_1}$ on variables $\vec{X}, \vec{Y}$ and for every constraint application C_j on variables $\vec{V}_j$ (which is a subset of variables from $\vec{X}, \vec{Y}$) we place the constraints $C'_{1,j}, \ldots, C'_{m_2,j}$ on variable set $\vec{V}_j, \vec{Z}'_j$ with $\vec{Z}'_j$ being the auxiliary variables.

We now show that this collection of constraints satisifies properties (a)-(c) with $\alpha = \alpha_1 + \beta(\alpha_2 - 1)$. Additionally we show that perfectness and/or strictness is preserved. We start with properties (a) and (c).

Consider any assignment to $\vec{X}$ satisfying f. Then any assignment to $\vec{Y}$ satisfies at most α_1 constraints from the set $C_1, \ldots, C_{m_1}$. Let γ of these be from the set $C_1, \ldots, C_\beta$. Now for every $j \in \{1, \ldots, \beta\}$ any assignment to $\vec{Z}'_j$ satisfies at most α_2 of the constraints $C'_{1,j}, \ldots, C'_{m_2,j}$. Furthermore if the constraint C_j was not satisfied by the assignment to $\vec{X}, \vec{Y}$, then at most $\alpha_2 - 1$ constraints are satisfied. Thus the total number of constraints satisfied by any assignment is at most $\gamma(\alpha_2) + (\beta - \gamma)(\alpha_2 - 1) + (\alpha_1 - \gamma) = \alpha_1 + \beta(\alpha_2 - 1)$. This yields property (a). Property (c) is achieved similarly.

We now show that if the α_1- and α_2-implementations are perfect we get property (b) with perfectness. In this case for any assignment to $\vec{X}$ satisfying f, there exists an assignment to $\vec{Y}$ satisfying $C_1, \ldots, C_{m_1}$. Furthermore for every $j \in \{1, \ldots, \beta\}$, there exists an assignments to $\vec{Z}'_j$ satisfying all the constraints $C'_{1,j}, \ldots, C'_{m_2,j}$. Thus there exists an assignment to $\vec{X}, \vec{Y}, \vec{Z}'_1, \ldots, \vec{Z}'_\beta$ satisfying all $m_1 + \beta(m_2 - 1)$ constraints. This yields property (b) with perfectness.

We now consider the case when the α_1- and α_2-implementations satisfy property (d) and show that in this case also the collection of constraints above satisfies property (b). Given an assignment to $\vec{X}$ satisfying f there exists an assignment to $\vec{Y}$ satisfying α_1 constraints from $C_1, \ldots, C_{m_1}$. Say this assigment satisfied γ clauses from the set $C_1, \ldots, C_\beta$ and $\alpha_1 - \gamma$ constraints from the set $C_{\beta+1}, \ldots, C_{m_1}$. Then for every $j \in \{1, \ldots, \beta\}$ such that the clauses C_j is satisfied by this assignment to $\vec{X}, \vec{Y}$, there exists an assignment to $\vec{Z'}_j$ satisfying α_2 clauses from the set $C'_{1,j}, \ldots, C'_{m_2,j}$. Furthermore, for the remaining values of $j \in \{1, \ldots, \beta\}$ there exists an assignment to the variables $\vec{Z'}_j$ satisfying $\alpha_2 - 1$ of the constraints $C'_{1,j}, \ldots, C'_{m_2,j}$ (here we are using the strictness of the α_2 implementations). This setting to $\vec{Y}, \vec{Z'}_1, \ldots, \vec{Z'}_\beta$ satisfies $\gamma\alpha_2 + (\beta - \gamma)(\alpha_2 - 1) + \alpha_1 - \gamma = \alpha_1 + \beta(\alpha_2 - 1)$ of the m constraints. This yields property (b). A similar analysis can be used to show property (d). ■

We next show a simple monotonicity property of implementations.

Lemma 3.3 For integers α, α' with $\alpha \leq \alpha'$, if $\mathcal{F}$ α-implements f then $\mathcal{F}$ α'-implements f. Furthermore strictness and perfectness are preserved under this transformation.

Proof: Let constraint applications $C_1, \ldots, C_m$ from $\mathcal{F}$ on $\vec{X}, \vec{Y}$ form an α-implementation of f. Let g be a constraint from $\mathcal{F}$ that is satisfiable and let k be the arity of g. Let $C_{m+1}, \ldots, C_{m+\alpha'-\alpha}$ be $\alpha' - \alpha$ applications of the constraint g on new variables $Z_1, \ldots, Z_k$. Then the collection of constraints $C_1, \ldots, C_{m+\alpha'-\alpha}$ on variable set $\vec{X}, \vec{Y}, \vec{Z}$ form an α'-implementation of f. Furthermore the transformation preserves strictness and perfectness. ■

The next two lemmas show how we can use implementations to prove the hardness of MAX CSP and weighted MAX ONE problems respectively.

Lemma 3.4 If MAX CSP($\mathcal{F}_1$) is APX-hard and every constraint of $\mathcal{F}_1$ can be strictly implemented by a constraint of $\mathcal{F}_2$, then MAX CSP($\mathcal{F}_2$) is APX-hard.

Proof: Assume without loss of generality that every constraint in $\mathcal{F}_1$ is satisfiable (i.e., $\mathcal{F}'_1 = \mathcal{F}_1$). Then for every constraint $f \in \mathcal{F}_1$ of arity k, an assignment s chosen uniformly from $\{0,1\}^k$ is a satisfying assignment with probability at least $1/2^k$. Let L be the maximum over all $f \in \mathcal{F}_1$ of 2^{k_f} where k_f is the arity of f. Then notice that for any instance $\mathcal{I}$ with m constraints, the value of the optimum is at least m/L.

Let α be a postive integer such that there exists a strict α-implementation of every constraint in $\mathcal{F}_1$ using $\mathcal{F}_2$; notice that the Lemma 3.3 implies that such an α does exist. Suppose there exists a β-approximation algorithm for MAX CSP($\mathcal{F}_2$) then we will show that there exists a β'-approximation algorithm for $\mathcal{F}_1$, where $\beta' \to 1$ as $\beta \to 1$. We focus on the case when $\beta < 1 + \frac{1}{(\alpha-1)L}$. For larger values of β we produce an L-approximate solution by using a random assignment to variables.

Given an instance $\mathcal{I}$ of MAX CSP($\mathcal{F}_1$) with m constraints and optimum o, we replace every constraint of $\mathcal{I}$ by its strict implementation using the constraints of $\mathcal{F}_2$. This produces an instance $\mathcal{I}'$ of MAX CSP($\mathcal{F}_2$) with optimum $o + (\alpha - 1)m$. Given a solution which satisfies m' of these constraints, the projection on to the variables of $\mathcal{I}$ satisfies at least $m' - (\alpha - 1)m$ constraints of $\mathcal{I}$. Thus if this solution is the output of the β-approximation algorithm (i.e., $m' \geq \frac{1}{\beta}(o + (\alpha - 1)m)$), then this yields a solution which is a β'-approximate solution to $\mathcal{I}$, where β' is bounded by:

$$\begin{aligned}
\beta' &\leq \frac{o}{\frac{1}{\beta}((\alpha-1)m + o) - (\alpha-1)m} \\
&= \frac{1}{\frac{1}{\beta} - (\alpha-1)(1-\frac{1}{\beta})(\frac{m}{o})} \\
&\leq \frac{\beta}{1 - (\alpha-1)(\beta-1)L}
\end{aligned}$$

Notice that the denominator is positive if $\beta < 1 + \frac{1}{(\alpha-1)L}$. Further, if $\beta = 1+\epsilon$, then β' is bounded by $1+\epsilon\left(\frac{1+(\alpha-1)L}{(1-(\alpha-1)\epsilon L}\right)$ and thus $\beta' \to 1$ and $\beta \to 1$. Thus a β-approximation algorithm for MAX CSP($\mathcal{F}_2$) yields a β'-approximation algorithm for MAX CSP($\mathcal{F}_1$), where $\beta' = \frac{\beta}{1-(\alpha-1)(\beta-1)L}$. It is easy to see that $\beta' \to 1$ as $\beta \to 1$ and thus the APX-hardness of MAX CSP($\mathcal{F}_1$) implies the APX-hardness of MAX CSP($\mathcal{F}_2$). ■

Lemma 3.5 Given constraint sets $\mathcal{F}_1, \mathcal{F}_2$, such that the weighted MAX ONE($\mathcal{F}_2$) problem has a $\alpha(n)$-approximation algorithm and every constraint $f \in \mathcal{F}_1$ can be perfectly implemented by the constraint set $\mathcal{F}_2$, then there exist constants c, d such that the weighted MAX ONE($\mathcal{F}_1$) problem has a $\alpha(cn^d)$-approximation algorithm.

Proof: Let $l = |\mathcal{F}_1|$ and k be the maximum arity of any constraint $f \in \mathcal{F}_1$. Let K be the largest number of auxiliary variables used in perfectly implementing any constraint $f \in \mathcal{F}_1$ by $\mathcal{F}_2$. Notice that K is a finite constant for any fixed $\mathcal{F}_1$, $\mathcal{F}_2$.

Given an instance $\mathcal{I}$ of MAX ONE($\mathcal{F}_1$) with m constraints $C_1, \ldots, C_m$ on n variables $X_1, \ldots, X_n$, with n real non-negative weights $w_1, \ldots, w_n$, we create an instance $\mathcal{I}'$ of MAX ONE($\mathcal{F}_2$) as follows: $\mathcal{I}'$ has the variables $X_1, \ldots, X_n$ of $\mathcal{I}$ and in addition "auxiliary" variables $\{Y_i^j\}_{i=1,j=1}^{m,K}$. The weights corresponding to $X_1, \ldots, X_n$ is $w_1, \ldots, w_n$ (same as in $\mathcal{I}$) and the auxiliary variables Y_i^j have weight zero. The constraints of $\mathcal{I}'$ perfectly implement the constraints of $\mathcal{I}$. In particular the constraint $f_i(X_{i_1}, \ldots, X_{i_k})$ of $\mathcal{I}$ is implemented by a collection of constraints from $\mathcal{F}_2$ (as dictated by the perfect implementation of f_i by $\mathcal{F}_2$) on the variables $(X_{i_1}, \ldots, X_{i_k}, Y_i^1, \ldots, Y_i^K)$.

By the definition of perfect implementations, it is clear that the every feasible solution to $\mathcal{I}$ can be extended (by some assignment to the Y variables) into a feasible solution to $\mathcal{I}'$. Alternately, every solution to $\mathcal{I}'$ immediately projects on to a solution of $\mathcal{I}$. Furthermore, the value of the objective function is exactly the same (by our choice of weights). Thus a β-approximate solution to $\mathcal{I}'$ gives a β-approximate solution to $\mathcal{I}$.

It remains to study this approximation as a function of the instance size. Observe that the instance size of $\mathcal{I}'$ is much larger. Let N denote the number of variables in $\mathcal{I}'$. Then N is upper bounded by $Km + n$, where m is the number of constraints in $\mathcal{I}$. But m, in turn, is at most ln^k. Thus $N \le (K+1)ln^k$, implying that an $\alpha(N)$-approximate solution to $\mathcal{I}'$, gives an $\alpha((K+1)ln^k)$-approximate solution to $\mathcal{I}$. Thus an $\alpha(N)$-approximation algorithm for the weighted MAX ONE($\mathcal{F}_2$) problem yields an $\alpha(cn^d)$-approximation for the weighted MAX ONE($\mathcal{F}_1$)-problem, for $c = (K+1)l$ and $d = k$. ■

Corollary 3.6 If weighted MAX ONE($\mathcal{F}_1$) is APX-hard and $\mathcal{F}_2$ perfectly implements every constraint in $\mathcal{F}_1$, then weighted MAX ONE($\mathcal{F}_2$) is APX-hard. Similarly, if weighted MAX ONE($\mathcal{F}_1$) is poly-APX-hard and $\mathcal{F}_2$ perfectly implements every constraint in $\mathcal{F}_1$, then weighted MAX ONE($\mathcal{F}_2$) is poly-APX-hard.

4 The Classification Theorem for MAX CSP

In this section, we prove Theorem 2.2. Section 4.1 discusses the cases for which MAX CSP($\mathcal{F}$) is in P, while Section 4.2 proves APX-completeness for the remaining cases. Sections 4.3 and 4.4 show how some of the results derived in this section can be used to strengthen Schaefer's original theorem.

4.1 Polynomial Time Solvability

From this subsection onwards, we omit the notation $\mathcal{F}'$ (as discussed in Section 2) and assume that we have a constraint set $\mathcal{F}$ such that $\mathcal{F} = \mathcal{F}'$.

Lemma 4.1 The problem weighted MAX CSP($\mathcal{F}$) is in P if each $f_i \in \mathcal{F}$ is 0-valid.

Proof: Set each variable to zero; this satisfies all the constraints. ∎

Lemma 4.2 The problem weighted MAX CSP($\mathcal{F}$) is in P if each $f_i \in \mathcal{F}$ is 1-valid.

Proof: Set each variable to one; this satisfies all the constraints. ∎

Lemma 4.3 The problem weighted MAX CSP($\mathcal{F}$) is in P if each f_i is a 2-monotone constraint.

Proof: We reduce the problem of finding the maximum number of satisfiable constraints to the problem of finding the minimum number of unsatisfied constraints. This problem, in turn, reduces to the problem of finding s-t min-cut in directed graphs. 2-monotone constraints have the following possible forms : (a) $X_{i_1}X_{i_2}...X_{i_p}$, (b) $\overline{X_{j_1}}\ \overline{X_{j_2}}...\overline{X_{j_q}}$, and (c) $X_{i_1}X_{i_2}...X_{i_p} + \overline{X_{j_1}}\ \overline{X_{j_2}}...\overline{X_{j_q}}$ where $p, q \geq 1$.

Construct a directed graph G with two special nodes F and T and a vertex X_i corresponding to each variable in the input instance. Let ∞ denote an integer larger than the total weight of constraints. Now we proceed as follows for each of the above classes of constraints :

- For a constraint C of weight w of the form (a), create a new node e_C and add an edge from each X_i to e_C of cost ∞ and an edge from e_C to T of cost w.
- For a constraint C of weight w of the form (b), create a new node $\overline{e_C}$ and add an edge of cost ∞ from $\overline{e_C}$ to each X_i and an edge from F to $\overline{e_C}$ of cost w.
- Finally, for a constraint C of weight w of the form (c), we create two nodes e_C and $\overline{e_C}$ and connect e_C to $X_{i_1}, X_{i_2}, \ldots$ and connect $\overline{e_C}$ to $X_{j_1}, X_{j_2}, \ldots$ as described above and replace the edges of cost w from F and to T by an edge from e_C to $\overline{e_C}$ of cost w.

Using the correspondence between cuts and assignments which places vertices corresponding to true variables on the T side of the cut, we find that the cost of a minimum cut separating T from F equals the minimum weight of constraints that can be left unsatisfied. ∎

The next lemma shows that s-t min-cut problem is in weighted MAX CSP($\mathcal{F}$) for some 2-monotone constraint set $\mathcal{F}$. Since the previous lemma shows how to solve MAX CSP($\mathcal{F}$) for any 2-monotone constraint set $\mathcal{F}$ by reduction to s-t min-cut problem, it seems that s-t min-cut problem is the hardest (and perhaps the only interesting) problem in MAX CSP($\mathcal{F}$) for constraint sets $\mathcal{F}$ that are 2-monotone.

Lemma 4.4 The s-t min-cut problem is in weighted MAX CSP($\mathcal{F}$) for some 2-monotone constraint set $\mathcal{F}$.

Proof: Let $\mathcal{F}$ be a constraint set of three constraints $\{f_1, f_2, f_3\}$ such that $f_1(X) = \bar{X}$, $f_2(X, Y) = X + \bar{Y}$ and $f_3(Y) = Y$.

Now given an instance $G = (V, E)$ to s-t min-cut problem, we construct an instance of MAX CSP($\mathcal{F}$) on variables $X_1, X_2, ..., X_n$ where X_i corresponds to the vertex $X_i \in V$:

- For each edge $e = (s, x)$ with weight w_e, we create the constraint $f_1(X)$ with weight w_e.

- For each edge $e = (x, t)$ with weight w_e, we create the constraint $f_3(X)$ with weight w_e.
- For each edge $e = (x, y)$ with weight w_e and such that $x, y \notin \{s, t\}$, we create the constraint $f_2(X, Y)$ with weight w_e.

Given a solution to this instance of MAX CSP($\mathcal{F}$), we construct an s-t cut by placing the vertices corresponding to the false variables on the s-side of the cut and the remaining on the t-side of the cut. It is easy to verify that an edge e contributes to the cut iff its corresponding constraint is unsatisfied. Hence the optimal MAX CSP($\mathcal{F}$) solution and the optimal s-t min-cut solution coincide. ∎

4.2 Proof of APX-completeness

In this section we prove that for every $\mathcal{F}$ that is not 0-valid or 1-valid or 2-monotone, the problem MAX CSP($\mathcal{F}$) is APX-complete. The proof of containment in APX follows from the fact that MAX CSP is contained in MAX SNP which was shown to be in APX by Papadimitriou and Yannakakis [13]. It remains to show that that a constraint set which is not entirely 0-valid or entirely 1-valid or entirely 2-monotone gives a APX-hard problem. The main APX-hard problem which we reduce to any of these new ones is the MAX CUT problem shown to be hard by the results of Papadimitriou and Yannakakis [13] and Arora et al. [1]. Initially (in Lemma 4.13) we consider the case where we are essentially allowed to repeat variables and set some variables to true or false. This provides a relatively painless proof that if a constraint is not 2-monotone, then it provides a APX-hard problem. We then use the availability of constraints that are not 0-valid or 1-valid to strictly implement constraints which force variables to be 1 and 0 respectively, as well as to force variables to be equal. This eventually allows us to use Lemma 4.13. We first start with some notation.

4.2.1 Notation

Given an assignment s to an underlying set of variables, $Z(s)$ denotes the set of positions corresponding to variables set to zero and $O(s)$ denotes the set of positions corresponding to variables set to one. More formally, given an assignment $s = s_1 s_2 ... s_n$ to $X_1, X_2, ..., X_n$, where $s_i \in \{0, 1\}$, we have $Z(s) = \{i \mid s_i = 0\}$ and $O(s) = \{i \mid s_i = 1\}$.

Definition 4.5 [Unary Constraints] The constraints $T(X) = X$ and $F(X) = \bar{X}$ are called *unary constraints.*

Definition 4.6 [XOR and REP Constraints] The constraint $f(X, Y) = X \oplus Y$ is called the XOR constraint and its complement constraint, namely $X = Y$, is called the REP constraint.

Definition 4.7 [C-closed constraint] A constraint f is called *C-closed (or complementation-closed)* if for all assignments s, $f(s) = f(\bar{s})$.

Definition 4.8 [0/1-Consistent Set] A set $V \in \{1, \ldots, k\}$ is 0-*consistent* (1-*consistent)* for a constraint $f : \{0,1\}^k \to \{0,1\}$ if and only if every assignment s with $Z(s) \supset V$ (resp. $O(s) \supset V$) is a satisfying assignment for f.

Lemma 4.9 MAX CSP(XOR) is APX-hard.

Proof: We observe that MAX CSP(XOR) captures the MAX CUT problem shown to be APX-hard by [13, 1]. Given a graph $G = (V, E)$ with n vertices and m edges, create an instance $\mathcal{I}_G$ of MAX CSP($\mathcal{F}$) with one variable X_u for every vertex $u \in V$ and with constraints $\mathrm{XOR}(X_u, X_v)$ corresponding to every edge $\{u, v\} \in E$. It is easily seen there is a one-to-one correspondence between (ordered) cuts in G and the assignments to the variables of $\mathcal{I}_G$ which maintains the values of the objective functions (i.e., the cut value and the number of satisfied constraints). ■

Lemma 4.10 For any $f \in \{X+Y, X\bar{Y}, \bar{X}+\bar{Y}\}$, $\{f, T, F\}$ strictly implements the XOR constraint.

Proof: If $f = X+Y$, then the instance $\{f(X,Y), f(X,Y), F(X), F(Y)\}$ is a strict 3-implementation of $X \oplus Y$; if $f = X\bar{Y}$, then the instance $\{f(X,Y), f(Y,X)\}$ is a strict 1-implementation of $X \oplus Y$; and finally, if $f = \bar{X} + \bar{Y}$, then $\{f(X,Y), f(X,Y), T(X), T(Y)\}$ is a strict 3-implementation of $X \oplus Y$. ■

Lemma 4.11 $\{f, T, F\}$ strictly implements the REP constraint if f is the constraint $\bar{X} + Y$.

Proof: The instance $\{f(X,Y), f(X,Y), F(X), T(Y)\}$ is a strict 3-implementation of the constraint REP. ■

4.2.2 Characterizing 2-Monotone Constraints

In order to prove the hardness of a constraint which is not 2-monotone, we require to identify some characteristics of such constraints. The following gives a characterization that turns out to be useful.

Lemma 4.12 A constraint f is a 2-monotone constraint if and only if all the following conditions are satisfied:

(a) for every satisfying assignment s of f either $Z(s)$ is 0-consistent or $O(s)$ is 1-consistent.

(b) if V_1 is 1-consistent and V_2 is 1-consistent for f, then $V_1 \cap V_2$ is 1-consistent, and

(c) if V_1 is 0-consistent and V_2 is 0-consistent for f, then $V_1 \cap V_2$ is 0-consistent.

Proof: We use the fact that a constraint can be expressed in DNF form as a disjunction of conjunctions (sum of terms). For a 2-monotone constraint this implies that we can express it as a sum of two terms. Every satisfying assignment must satisfy one of the two terms and this gives Property (a). Properties (b) and (c) are obtained from the fact that the constraint has at most one term with all positive literals and at most one term with all negative literals.

Conversely consider a constraint f which satisfies properties (a)-(c). Let $s_1, \ldots, s_l$ be the satisfying assignments of f such that $Z(s_i)$ is 0-consistent, for $i \in \{1, \ldots, l\}$. Let $t_1 \ldots, t_k$ be the satisfying assignments of f such that $O(t_j)$ is 1-consistent, for $j \in \{1, \ldots, k\}$. Then $Z = \cap_i Z(s_i)$ and $O = \cap_j O(t_j)$ are 0-consistent and 1 consistent sets for f respectively (using (b) and (c)) which cover all satisfying assignments of f. Thus $f(\vec{X}) = (\wedge_{i \in Z} \bar{X}_i) \vee (\wedge_{j \in O} X_j)$, which is 2-monotone. ■

Observe that a 2-monotone constraint is always either 0-valid or 1-valid or both.

4.2.3 MAX SNP-hardness of Non 2-Monotone Constraints

We now use the characterization from the previous subsubsection to show that if one is allowed to "force" constants or "repetition" of variables, then the presence of non-2-monotone constraint gives hard problems. Rather than using the ability to force constants and repetitions as a binding requirement, we use them as additional constraints to be counted as part of the objective function. This is helpful later, when we try to remove the use of these constraints.

Lemma 4.13 For any constraint f which is not 2-monotone, $\{f, T, F, \mathrm{REP}\}$ strictly implements the constraint XOR.

Proof: We prove this by using the Lemma 4.12 for 2-monotone constraints. Let k denote the arity of f. If f is not 2-monotone, it must violate one of the three conditions (a), (b) and (c) stated in the Lemma 4.12.

Suppose f violates the property (a) of Lemma 4.12. Then for some satisfying assignment s, there exist two assignments s_0 and s_1 such that $Z(s) \subset Z(s_0)$ and $O(s) \subset O(s_1)$, but $f(s_0) = f(s_1) = 0$. Without loss of generality, we assume that $s = 0^p 1^q$, $s_0 = 0^{p+a} 1^{q-a}$ and $s_1 = 0^{p-b} 1^{q+b}$, where $p + q = k$. Thus we have the following situation :

$$\begin{array}{cccccc}
 & & & & & f() \\
s & \overbrace{00...0}^{p-a} & \overbrace{00...0}^{a} & \overbrace{11...1}^{b} & \overbrace{11...1}^{q-b} & 1 \\
s_0 & 00...0 & 00...0 & 00...0 & 11...1 & 0 \\
s_1 & 00...0 & 11...1 & 11...1 & 11...1 & 0 \\
s_2 & 00...0 & 11...1 & 00...0 & 11...1 & -
\end{array}$$

Observe that both a and b are non-zero. Consider the following set of constraints on variables $X_1, X_2, ..., X_k$:

- constraints $F(X_i)$ for $1 \le i \le (p - a)$,
- constraints $\mathrm{REP}(X_{p-a+1}, X_{p-a+i})$ for $2 \le i \le a$,
- constraints $\mathrm{REP}(X_{p+1}, X_{p+i})$ for $2 \le i \le b$,
- constraints $T(X_i)$ for $(p + a + b + 1) \le i \le k$, and
- the constraint $f(X_1, X_2, ..., X_k)$.

It is easy to verify that for $\alpha = (k - 1)$, this is a strict implementation of the constraint $X_{p-a+1} \oplus X_{p+1}$ if $f(s_2) = 1$ and $X_{p-a+1}\overline{X_{p+1}}$, otherwise. The claim now follows immediately from Lemma 4.10.

Next suppose f violates the property (b) of Lemma 4.12. Then there exists an unsatisfying assignment s to f satisfying $V_1 \cap V_2 \subset O(s)$. Notice that $O(s)$ cannot contain V_1 or V_2 (since V_1 and V_2 are 1-consistent for f). Thus s sets all variables in $V_1 \cap V_2$ to 1, and at least one variable in each of $V_1 \setminus (V_1 \cap V_2)$ and $V_2 \setminus (V_1 \cap V_2)$ to 0. Consider one such unsatisfying assignment s. Without loss of generality, we have the following situation :

$$\begin{array}{cccccccc}
 & \multicolumn{3}{c}{\overbrace{\hspace{12em}}^{V_1}} & & & & \\
 & & & \multicolumn{3}{c}{\overbrace{\hspace{12em}}^{V_2}} & & \\
s & \underbrace{\overbrace{00...0}^{V_1\setminus O(s)}}_{p} & \underbrace{11...1}_{q} & \underbrace{\overbrace{11...1}^{V_1\cap V_2}}_{r} & \underbrace{11...1}_{t} & \underbrace{\overbrace{00...0}^{V_2\setminus O(s)}}_{u} & \underbrace{00...0}_{v} & \underbrace{11...1}_{w}
\end{array}$$

Consider the following set of constraints on variables $X_1, X_2, ..., X_k$:

- constraints $\mathrm{REP}(X_1, X_i)$ for $2 \le i \le p$,
- constraints $T(X_i)$ for $(p + 1) \le i \le (p + q + r + t)$,
- constraints $\mathrm{REP}(X_{p+q+r+t+1}, X_{p+q+r+t+i})$ for $2 \le i \le u$,

- constraints $F(X_i)$ for $(p+q+r+t+u+1) \leq i \leq (p+q+r+t+u+v)$,
- constraints $T(X_i)$ for $(p+q+r+t+u+v) \leq i \leq (p+q+r+t+u+v+w)$, and finally
- the constraint $f(X_1, X_2, ..., X_k)$ where $k = (p+q+r+t+u+v+w)$.

It is now easy to verify that for $\alpha = (k-1)$, this is a strict α-implementation of the constraint $X_1 + X_{p+q+r+t+1}$. Again, the claim now follows immediately from Lemma 4.10.

Finally, the case in which f violates the property (c) above, can be handled in an analogous manner. ■

4.2.4 Implementing the REP Constraint

We now start on the goal of removing the use of the unary and replication constraints above. In order to do so we use the fact that we have available to us constraints which are not 0-valid and not 1-valid. It turns out that the case in which the same constraint is not 0-valid and not 1-valid and further has the property that its behavior is closed under complementation (i.e., $f(s) = f(\bar{s})$) is somewhat special. We start by analyzing this case first.

Lemma 4.14 Let f be a non-trivial constraint which is C-closed and is neither 0-valid nor 1-valid. Then $\{f\}$ perfectly and strictly implements the XOR constraint.

Proof: Let k denote the arity of f and let k_0 and k_1 respectively denote the maximum number of 0's and 1's in any satisfying assignment for f; clearly $k_0 = k_1$. Now let $S_X = \{X_1, X_2, ..., X_{2k}\}$ and $S_Y = \{Y_1, Y_2, ..., Y_{2k}\}$ be two disjoint sets of $2k$ variables each. We begin by placing the constraint f on a large collection of inputs as follows: for each satisfying assignment s, we place $\binom{2k}{i}\binom{2k}{k-i}$ constraints on the variable set $S_X \cup S_Y$ such that every i-variable subset of S_X appears in place of 0's in s and every $(k-i)$ variable subset of S_Y appears in place of 1's in the assignment s, where i denotes the number of 0's in s. Let this collection of constraints be denoted by $\mathcal{I}$.

Clearly, any solution which assigns identical values to all variables in S_X and the complementary value to all variables in S_Y, satisfies all the constraints in $\mathcal{I}$. We wish to show the converse, i.e., every assignment satisfying all the above constraints assigns identical values to all variables in S_X and the complementary value to every variable in S_Y.

Let Z and O respectively denote the set of variables set to zero and one respectively. We claim that any solution which satisfies all the constraints must satisfy either $Z = S_X$ or $Z = S_Y$.

To see this, assume without loss of generality that $|S_X \cap Z| \geq k$. This implies that $|S_Y \cap O| \geq k$ or else there exists a constraint in $\mathcal{I}$ with all its input variables set to zero and hence is unsatisfied. This in turn implies that no variable in S_X can take value one; otherwise, there exists a constraint with $k_1 + 1$ of its inputs set to one, and is therefore unsatisfied. Finally, we can now conclude that no variable in S_Y takes value zero; otherwise, there exists a constraint with $k_0 + 1$ of its inputs set to zero and is therefore unsatisfied. Thus, $Z = S_X$. Analogously, we could have started with the assumption that $|S_X \cap O| \geq k$ and established $Z = S_Y$. Hence an assignment satisfies all the constraints in $\mathcal{I}$ iff it satisfies either the condition $Z = S_X$ or the condition $Z = S_Y$.

We now augment the collection of constraints as follows. Consider a least hamming weight satisfying assignment s for f. Without loss of generality, we assume that $s = 10^p1^q$. Clearly then, $s' = 0^{p+1}1^q$ is not a satisfying assignment. Since f is C-closed, we have the following situation :

$$
\begin{array}{lcccc}
 & & \overbrace{}^{p} & \overbrace{}^{q} & f() \\
s' & 0 & 00...0 & 11...1 & 0 \\
s & 1 & 00...0 & 11...1 & 1 \\
\bar{s} & 0 & 11...1 & 00...0 & 1 \\
\bar{s'} & 1 & 11...1 & 00...0 & 0
\end{array}
$$

We add the constraints $f(X, X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_q)$ and $f(Y, Y_1, Y_2, ..., Y_p, X_1, X_2, ..., X_q)$. If $X = 1$, then to satisfy the constraint $f(X, X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_q)$, we must have $Z = S_X$. Otherwise, we have $X = 0$ and then to satisfy the constraint $f(X, X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_q)$ we must have $Z = S_Y$. In either case, the only way we can also satisfy the constraint

$$f(Y, Y_1, Y_2, ..., Y_p, X_1, X_2, ..., X_q)$$

is by assigning Y the complementary value. Thus these set of constraints perfectly and strictly implement the constraint $X \oplus Y$; all constraints can be satisfied iff $X \neq Y$ and if $X = Y$ there exists an assignment to variables in S_X and S_Y such that precisely 1 constraint is unsatisfied.

■

4.2.5 Implementing the Unary Constraints

If the constraint(s) which is (are) not 0-valid and 1-valid is (are) not closed under complementation, then they can be used to get rid of the unary constraints. This is shown in the next lemma.

Lemma 4.15 Let f_0 and f_1 be two non-trivial constraints, possibly identical, which are not 0-valid and 1-valid respectively. If neither f_0 nor f_1 is C-closed, then $\{f_0, f_1\}$ perfectly and strictly implement both the unary constraints $T(.)$ and $F(.)$.

Proof: We will only describe the implementation of constraint $T(.)$; the analysis for the constraint $F(.)$ is identical. Assume, for simplicity, that both f_i, $i \in \{0, 1\}$, are of arity k. We build on the implementation in the proof of Lemma 4.14. We start with a set of $4k$ variables $S_X = \{X_1, \ldots, X_{2k}\}$ and $S_Y = \{Y_1, \ldots, Y_{2k}\}$. For each $i \in \{0, 1\}$, for each satisfying assignment s of f_i, if j is the number of 0's in s we place the $\binom{2k}{j}\binom{2k}{k-j}$ constraints f_i with all possible subsets of S_X appearing as the zero set and all possible subsets of S_Y appearing as the one set. Now we argue that any solution which satisfies all the constraints in $\mathcal{I}_0$ and $\mathcal{I}_1$ must set all variables in S_X to 0 and all variables in S_Y to 1.

So we have two constraints f_0 and f_1 such that neither is C-closed. Suppose $|S_X \cap O| \geq k$ then we must have $|S_Y \cap O| \geq k$. To see this, consider a satisfying assignment s such that $f_0(\bar{s}) = 0$; there must exist such an assignment since f_0 is not C-closed. Now if $|S_Y \cap Z| \geq k$, then clearly at least one constraint corresponding to s is unsatisfied - namely, the one in which the positions in $O(s)$ are occupied by the variables in $(S_Y \cap Z)$ and the positions in $Z(s)$ are occupied by the variables in $(S_X \cap O)$. Thus we must have $|S_Y \cap O| \geq k$. But if we have both $|S_X \cap O| \geq k$ and $|S_Y \cap O| \geq k$, then there is at least one unsatisfied constraint in the instance $\mathcal{I}_1$ since f_1 is not 1-valid. Thus this case cannot arise.

So we now consider the case $|S_X \cap Z| \geq k$. Then for constraints in $\mathcal{I}_0$ to be satisfied, we must once again have $|S_Y \cap O| \geq k$; else there is a constraint with all its inputs set to zero and is hence unsatisfied. This can now be used to conclude that $S_Y \cap Z = \phi$ as follows. Consider a satisfying assignment with smallest number of ones. This number is positive since f_0 is not 0-valid. If we consider all the constraints corresponding to this assignment with inputs from S_Y and $S_X \cap Z$ only, it is easy to see that there will be at least one unsatisfied constraint if $S_Y \cap Z \neq \phi$. Hence each variable in S_Y is set to one in this case. Finally, using the constraints on the constraint f_1 which is not 1-valid, it is easy to conclude that in fact $Z = S_X$.

Now let $s = 10^p1^q$ be a least hamming weight satisfying assignment for f_0; p, q may be zero but s contains at least a single one as f_0 is not 0-valid. Then the constraint $f_0(X, X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_q)$ can be satisfied iff $X = 1$. Thus all the constraints in $\mathcal{I}_0$ and $\mathcal{I}_1$ are satisfied along with above constraint iff $X = 1$ and otherwise, we can still satisfy all the constraints in $\mathcal{I}_0$ and $\mathcal{I}_1$. Hence this is indeed an implementation of the constraint $T(.)$. The constraint $F(.)$ can be implemented in an analogous manner.

■

4.2.6 Unary Constraints Help Implement either REP or MAX SNP-Hard Constraints

Lemma 4.16 Let f be a constraint which is not 2-monotone. Then $\{f, T, F\}$ can strictly implement either the XOR or the REP constraint.

Proof: Since f is not 2-monotone and non-trivial, it must be sensitive to at least two variables. Consider the boolean k-cube with each vertex s labeled by the function value $f(s)$; where k is the arity of constraint f. Let V_i denote the set of vertices labeled i, $i \in \{0, 1\}$. If $|V_i| \leq |V_{1-i}|$, we claim that it must be the case that there exists a vertex in V_i which has at least two neighbors in V_{1-i}. This is readily seen using the expansion properties of the k-cube; any set S of at most 2^{k-1} vertices must have expansion factor at least one. Furthermore, the expansion factor is precisely one only when the set S induces a boolean $(k-1)$-cube. But the latter case can't arise since it would imply that f is a single variable constraint. Hence there must exist a vertex $s \in V_i$ which has two neighbors in V_{1-i}.

Let s_i and s_j be these two neighbors of s, differing in the $i^{\underline{\text{th}}}$ and the $j^{\underline{\text{th}}}$ bit position respectively. Without loss of generality, we may assume that $i = 1$ and $j = 2$. Consider now the input instance which has a constraint of the form $f(X_1, X_2, Y_3, Y_4, ..., Y_k)$ and constraints of the form $T(Y_i)$ for each i in $O(s) - \{1, 2\}$ and of the form $F(Y_i)$ for each i in $Z(s) - \{1, 2\}$. It is now easy to verify that this set of constraints strictly implement one of the constraints $X_1 \oplus X_2$, $\overline{X_1 \oplus X_2}$, $X_1 + X_2$, $\bar{X}_1 + \bar{X}_2$ or $\bar{X}_1 + X_2$. The first two are the constraints XOR and REP respectively. By Lemma 4.10, either of the constraints $X_1 + X_2$ or $\bar{X}_1 + \bar{X}_2$ along with $\{T, F\}$ strictly implement XOR, and by Lemma 4.11 the family $\{\bar{X}_1 + X_2, T, F\}$ strictly implements REP. ■

Lemma 4.17 If $\mathcal{F}$ is a constraint set such that there exist (1) $f_0 \in \mathcal{F}$ which is not 0-valid, (2) $f_1 \in \mathcal{F}$ which is not 1-valid and (3) $f_2 \in \mathcal{F}$ which is not 2-monotone, then MAX CSP($\mathcal{F}$) is MAX SNP-hard.

Proof: If either f_0 or f_1 is C-closed then using Lemma 4.14, we can strictly implement the XOR constraint.

If neither f_0 nor f_1 is C-closed, then using Lemma 4.15 $\{f_0, f_1, f_2\}$ strictly implements the unary constraints $T(.)$ and $F(.)$, and then using the Composition Lemma along with Lemma 4.16, we conclude that $\{f_0, f_1, f_2\}$ strictly implements either the XOR constraint or the REP constraint. In the latter case, we can use Lemma 4.13 to conclude that $\{f_0, f_1, f_2\}$ can strictly implement the XOR constraint.

Thus is every situation, $\mathcal{F}$ strictly implements the XOR constraint. The lemma follows from the fact that MAX CSP(XOR) is APX-hard (Lemma 4.9) and the approximation preserving property of strict implementations (Lemma 3.4). ■

4.3 Hardness at Gap Location 1

Schaefer's dichotomy theorem can be extended to show that in the cases where SAT($\mathcal{F}$) in NP-hard to decide, it is actually hard to distinguish satisfiable instances from instances which are not satisfiable in a constant fraction of the constraints. This is termed hardness at gap location 1 by Petrank [14] who highlights the usefulness of such hardness results in other reductions. The essential observation needed is that perfect implementations preserve hardness gaps located at 1 and that Schaefer's proof is based on perfect implementations. Thus we have the following theorem:

Theorem 4.18 For every constraint set $\mathcal{F}$ either SAT($\mathcal{F}$) is easy to decide, or there exists $\epsilon = \epsilon_{\mathcal{F}} > 0$ such that it is NP-hard to distinguish satisfiable instances of SAT($\mathcal{F}$), from instances where $1-\epsilon$ fraction of the constraints are not satisfiable.

4.4 Strengthening Schaefer's Dichotomy Theorem

Schaefer's proof of NP-hardness in his dichotomy theorem relies on the ability to replicate variables within a constraint application. We observe that this assumption can be eliminated by creating a perfect implementation of the function REP. Since given a perfect implementation, we can replace any p replicated copies of a variable X by p new variables $X_1, X_2, ..., X_p$ and add constraints of the form $\mathrm{REP}(X_1, X_2), \mathrm{REP}(X_1, X_3), ..., \mathrm{REP}(X_1, X_p)$. We now show how to create a perfect implementation of the REP function.

Lemmas 4.14 and 4.15 show that MAX CSP($\{f_0, f_1, f_2\}$), where f_0 is not 0-valid and f_1 is not 1-valid, can be used to create either a perfect implementation of the function REP or a perfect implementation of both unary functions T and F. In the latter case, we can show the following lemma.

Lemma 4.19 If f is not weakly negative then MAX CSP($\{f, T, F\}$) can perfect implement either the function $x \oplus y$, or the function $x + y$. Similarly, if f is not weakly positive then MAX CSP($\{f, T, F\}$) can perfect implement either the function $x \oplus y$, or the function $\bar{x} + \bar{y}$.

Proof: We only prove the first part - the second part follows by symmetry. We know that f has a maxterm S with at least two positive literals. We consider the function f' which is f existentially quantified over the variables not in S. Let x_1 and x_2 be the two positive literals in S. Set all other variables in S to the value which does not make S true. Then the assignment $x_1 = x_2 = 0$ is a non-satisfying assignment. The assignments $x_1 = 0 \neq x_2$ and $x_1 \neq 0 = x_2$ must be satisfying assignments ny the definition of maxterm. While the assignment $x_1 = x_2 = 1$ may go either way. Depending on this we get either the function $x \oplus y$ or $x + y$. ■

Corollary 4.20 If f_2 is not weakly positive and f_3 is not weakly negative, then MAX CSP($\{f_2, f_3, T, F\}$) perfectly implements (at gap 1) the XOR function.

Since the SAT($\mathcal{F}$) problems that we need to establish as NP-hard in Schaefer's theorem satisfy the condition that there exists $f_0, f_1, f_2, f_3 \in \mathcal{F}$ such that f_0 is not 0-valid and f_1 is not 1-valid, f_2 is not weakly positive and f_3 is not weakly negative, we conclude that we can perfectly implement the XOR function. This, in turn, can be used to create a perfect implementation of the function $\mathrm{REP}(x, y)$ by using the constraints $\{x \oplus z, y \oplus z\}$ for some auxiliary variable z. Thus replication can be eliminated from Schaefer's proof.

5 The Classification Theorem for MAX ONE

In this section, we establish Theorem 2.3, the MAX ONE classification theorem. We use the following shorthand notation for the eight constraint classes of importance. Let $\mathcal{F}_1$ denote the class of 1-valid constraints, $\mathcal{F}_2$ the weakly positive constraints, $\mathcal{F}_3$ the affine width-2 constraints, $\mathcal{F}_4$ the affine constraints, $\mathcal{F}_5$ the strongly 0-valid constraints, $\mathcal{F}_6$ the weakly negative constraints, $\mathcal{F}_7$ the 2CNF constraints and $\mathcal{F}_8$ the 0-valid ones. Theorem 2.3 can be restated as follows. For a constraint set $\mathcal{F}$ if i is the smallest index such that $\mathcal{F} \subset \mathcal{F}_i$, then if $i \in 1, 2, 3$ then MAX ONE($\mathcal{F}$) $\in$ P, if $i = 4$ then MAX ONE($\mathcal{F}$) is APX-complete, if $i \in 5, 6, 7$ then MAX ONE($\mathcal{F}$) is poly-APX-complete, if $i = 8$ then SAT($\mathcal{F}$) is in P but MAX ONE($\mathcal{F}$) is not approximable and if no such i exists then finding any satisfying assignment for MAX ONE($\mathcal{F}$) in NP-hard.

5.1 Preliminaries

In this subsection, we prove a few preliminary lemmas that we will need in the proof of the theorem, particularly in Cases 2 and 3. We first show that in these cases, it is essentially equivalent for us to consider the weighted or unweighted MAX ONE($\mathcal{F}$) problem.

We begin with a slightly stronger definition of polynomial-time solvability of SAT($\mathcal{F}$) that we will need. We then show that given this stronger form of SAT($\mathcal{F}$) that insofar as APX-hardness and poly-APX-hardness are concerned, the weighted and unweighted cases of MAX ONE($\mathcal{F}$) are equivalent. We conclude by showing that in Cases 2 and 3 the stronger form of SAT($\mathcal{F}$) holds.

Definition 5.1 We say that a constraint satisfaction problem SAT($\mathcal{F}$) is *strongly decidable* if given m constraints on n variables $X_1, \ldots, X_n$ *and an index* $i \in \{1, \ldots, n\}$, there exists a polynomial time algorithm which decides if there exists an assignment to $X_1, \ldots, X_n$ satisfying all m constraints and additionally satisfying the property $X_i = 1$.

Lemma 5.2 For every strongly decidable constraint set $\mathcal{F}$, for every ϵ of the form $1/l$ for some positive integer l and for every non-decreasing function $\alpha : \mathcal{Z}^+ \to \mathcal{Z}^+$, α-approximating the weighted MAX ONE($\mathcal{F}$) problem reduces to α'-approximating the (unweighted) MAX ONE($\mathcal{F}$) problem, where $\alpha'(n) = \frac{\alpha(\sqrt{\epsilon n})}{(1+\epsilon)}$.

Proof: We use the shorthand notation $(\vec{X}, \vec{C}, \vec{w})$ to denote an instnance of a weighted MAX ONE problem consisting of constraint applications $C_1, \ldots, C_m$ on variables $X_1, \ldots, X_n$ with weights $w_1, \ldots, w_n$. Given an instance $\mathcal{I} = (\vec{X}, \vec{C}, \vec{w})$ of a weighted MAX ONE($\mathcal{F}$) problem, we create a sequence of instances of weighted MAX ONE($\mathcal{F}$) problems with the final instance having $N \leq \frac{n^2}{\epsilon}$ variables, with the weight of every variable being 1, and with the property that given an $\alpha'(N)$-approximate solution to the final instance we can obtain an $\alpha(n)$-approximate solution to the instance I.

Assume w.l.o.g. that $w_1 \geq w_2 \geq \cdots \geq w_n$. From here onwards i will denote the smallest index such that there exists a feasible solution to $\mathcal{I}$ with $X_i = 1$. Notice that i can be computed in polynomial time. The instance $\mathcal{I}_1$ is defined to be $(\vec{X}, \vec{C}, \vec{w'})$, where $w'_j = w_j$ if $j \geq i$ and $w'_j = 0$ otherwise. Since no solution of $\mathcal{I}$ has $X_j = 1$ for $j < i$, this change in the weight function does not alter the value of any solution. Thus $\mathcal{I}_1$ is essentially equivalent to $\mathcal{I}$. Notice that the weight of the optimal solution to $\mathcal{I}_1$ is at least w'_i.

The instance $\mathcal{I}_2$ is defined to be $(\vec{X}, \vec{C}, \vec{w''})$, where w''_j is w'_j rounded up to the nearest integral positive multiple of $\frac{\epsilon w'_i}{n}$. Notice that the net increase to the weight of any solution by this increase in the weights is at most $\epsilon w'_i$.

Finally the (unweighted) instance $\mathcal{I}_3$ has as its variables N_j copies of every variable X_j, where $N_j = \frac{w''_j}{(\epsilon w'_i/n)}$. Notice that by the definition of w''_j's, the N_j's are integral and $1 \leq N_j \leq \frac{n}{\epsilon}$. Given a constraint C_i on variables $X_{j_1}, \ldots, X_{j_k}$, the instance $\mathcal{I}_3$ has $N_{j_1} \times N_{j_2} \times \cdots \times N_{j_k}$ copies of the constraint C_i applied to all different collections of k-tuples of variables containing a copy of the j_1th variable in the first position, j_2th variable in the second position and so on. The weight of all variables is 1.

Let $N \triangleq \sum_{j=1}^{n} N_j$ denote the number of variables in $\mathcal{I}_3$. We will now show that given $\alpha'(N)$-approximate solution to $\mathcal{I}_3$, we can reconstruct a $\alpha(n)$-approximate solution to $\mathcal{I}$. It is easy to verify that any solution to $\mathcal{I}_3$ can be modified to have all copies of a variable assigned to 1 or all assigned to zero, without changing any 1 to a 0. Thus every solution to $\mathcal{I}_2$ of weight w can be transformed to a solution of weight $\frac{w}{w_i'} \cdot \frac{n}{\epsilon}$ for $\mathcal{I}_3$ and vice versa. Thus given a solution of weight at least $\frac{\mathrm{OPT}(\mathcal{I}_3)}{\alpha'(N)}$, we can reconstruct (in polynomial time) a solution of weight at least $\frac{\mathrm{OPT}(\mathcal{I}_2)}{\alpha'(N)}$ for $\mathcal{I}_2$. To conclude the argument it suffices to show that this can be used to construct an $\alpha(n)$-approximate solution to $\mathcal{I}_1$.

Our candidate solutions will be the solution which assigns 1 to X_i and the solution to the instance $\mathcal{I}_2$. If the value of the $\alpha'(N)$-approximate solution to $\mathcal{I}_3$ is w, then the value of a solution so returned is at least $\max\{w_i', w - \epsilon w_i'\}$ which is at least $\frac{w}{1+\epsilon}$. (The inequality $\max\{w_i', w - \epsilon w_i'\} \geq \frac{w}{1+\epsilon}$ follows from a simple averaging argument.) Since $w \geq \mathrm{OPT}(\mathcal{I}_2)/\alpha'(N)$ and $\mathrm{OPT}(\mathcal{I}_1) \leq \mathrm{OPT}(\mathcal{I}_2)$, we find that this solution has value at least $\frac{\mathrm{OPT}(\mathcal{I}_1)}{(1+\epsilon)\alpha'(N)} = \frac{\mathrm{OPT}(\mathcal{I})}{(1+\epsilon)\alpha'(N)}$. Thus this solution is a $(1+\epsilon)\alpha'(N)$ approximate solution to $\mathcal{I}$.

Finally observe that since $N \leq \frac{n^2}{\epsilon}$, this is also an $(1+\epsilon)\alpha'(n^2/\epsilon) = \alpha(n)$-approximate solution to the instance $\mathcal{I}$. ∎

The ability to work with weighted problems in combination with Lemma 3.5 allows us to use existential quantification over auxiliary variables and the notion of perfect implementations of constraints.

As our examination will eventually show, there is really no essential difference in the approximability of the weighted and unweighted problems. For now we will satisfy ourselves by stating this conditionally.

Corollary 5.3 For any strongly decidable constraint set $\mathcal{F}$, the MAX ONE($\mathcal{F}$) problem is APX-hard if and only if the weighted MAX ONE($\mathcal{F}$) problem is APX-hard. Similarly, the MAX ONE($\mathcal{F}$) problem is poly-APX-hard if and only if the weighted MAX ONE($\mathcal{F}$) problem is poly-APX-hard.

Before concluding we show that most problems of interest to us will be able to use the equivalence between weighted and unweighted problems.

Lemma 5.4 If $\mathcal{F} \subset \mathcal{F}_j$ for some $j \in \{1, \ldots, 7\}$, then $\mathcal{F}$ is strongly decidable.

Proof: For a constraint $f \in \mathcal{F}$ and an index $i \in \{1, \ldots, k\}$, let f_i^* be the constraint:

$$f_i^*(X_1, \ldots, X_k) \stackrel{\text{def}}{=} f(X_1, \ldots, X_{i-1}, 1, X_{i+1}, \ldots, X_k).$$

Further let $\mathcal{F}^*$ be the constraint set:

$$\mathcal{F}^* \stackrel{\text{def}}{=} \mathcal{F} \cup \{f_i^* | f \in \mathcal{F}, i \in [k]\}.$$

First observe that the problem of strong decidability of $\mathcal{F}$ is the decision problem SAT($\mathcal{F}^*$). Further, observe that if $\mathcal{F} \subset \mathcal{F}_j$ for $j \in \{1, 2, 3, 4, 6, 7\}$, then $\mathcal{F}^* \subset \mathcal{F}_j$ as well. Lastly, if $\mathcal{F}^* \subset \mathcal{F}_5$, then $\mathcal{F}^* \subset \mathcal{F}_8$. Thus in each case we end up with a problem from SAT($\mathcal{F}$) which is in P by Schaefer's theorem. ∎

Lastly we describe one more tool that comes in useful in creating reductions. This is the notion of implementing a property which falls short of being an implementation of an actual constraint. The target constraints in the following definitions are the constraints which force variables to being constants (either 0 or 1). However, sometimes we are unable to achieve this. So we end up implementing a weaker form which however suffices for our applications. We next describe this property.

Definition 5.5 [Existential Zero] A constraint set $\mathcal{F}$ can implement the existential zero property if there exists a set of m constraints $f_1, \ldots, f_m$ over n variables $\vec{X}$ and an index $k \in \{1, \ldots, n\}$ such that the following hold:

- There exists an assignment $V_{k+1}, \ldots, V_n$ to $X_{k+1}, \ldots, X_n$ such that assigning 0 to the first k variables $X_1, \ldots, X_k$ satisfies all constraints.
- Conversely, every assignment satisfying all the constraints must make at least one of the variables in $X_1, \ldots, X_k$ zero.

An *Existential One* can be defined similarly.

Definition 5.6 Given a constraint f of arity k and a set $S \subset \{1, \ldots, k\}$, the constraint $f|_{(S,0)}$ is a constraint of arity $k - |S|$ given by $f|_{(S,0)}(X_1, \ldots, X_{k-|S|}) = f(X_1, 0, 0, X_2, \ldots, X_{k-|S|}, 0, 0)$, where the zeroes occur in the indices contained in S. For a constraint set $\mathcal{F}$, the 0-closure of $\mathcal{F}$, denoted $\mathcal{F}|_0$ is the set of constraints $\{f_{(S,0)} | f \in \mathcal{F}, S \subset \{1, \ldots, k\}\}$. (1-closure may be defined similarly.)

Notice that $\mathcal{F}|_0$ essentially implements every constraint that can be implemented by $\mathcal{F} \cup \{F\}$, except the constraint $\{F\}$, where F stands for the unary constraint "false". We define $\mathcal{F}|_1$ similarly. Then $\mathcal{F}|_{0,1} = \mathcal{F}|_0 \cup \mathcal{F}|_1$.

Lemma 5.7 If a constraint set $\mathcal{F}$ can implement the existential zero property, then $\mathcal{F}$ perfectly implements every constraint in the constraint set $\mathcal{F}|_0$. Similarly, if a constraint set $\mathcal{F}$ can implement the existential one property, then $\mathcal{F}$ perfectly implements every constraint in the constraint set $\mathcal{F}|_1$.

Proof: We show how to implement the constraint $f(0, X_1, \ldots, X_{k-1})$. The proof can be extended to other sets by induction. Let the constraints $f_1, \ldots, f_m$ implement the existential zero property on variables $Y_1, \ldots, Y_K$ with auxiliary variables $Y_{K+1}, \ldots, Y_N$. Then the constraints $f(Y_i, X_1, \ldots, X_{k-1})$, for $1 \leq i \leq K$, along with the constraints $f_1, \ldots, f_m$ on the variables $Y_1, \ldots, Y_N$ perfectly implement the constraint $f(0, X_1, \ldots, X_{k-1})$. (Observe that since at least one of the Y_i's in the set $Y_1, \ldots, Y_K$ is zero, the constraint $f(0, X_1, \ldots, X_{k-1})$ is being enforced.) Furthermore, we can always set all of $Y_1, \ldots, Y_K$ to zero, ensuring that any assignment to $X_1, \ldots, X_{k-1}$ satisfying $f(0, X_1, \ldots, X_{k-1})$ does satisfy all the constraints listed above. ■

5.2 Proof of Main Theorem

5.2.1 Case 1: The Polynomial Time Solvable Case

We now begin our proof of Theorem 2.3 with a relatively simple case.

Lemma 5.8 The weighted MAX ONE($\mathcal{F}$) problem is in P if $\mathcal{F}$ is 1-valid or is weakly positive or is affine with width 2.

Proof: If $\mathcal{F}$ is 1-valid, then setting each variable to 1 satisfies all constraint applications with the maximum possible variable weight.

If $\mathcal{F}$ is weakly positive, consider the CNF formulae for the $f_i \in \mathcal{F}$ such that each clause has at most one negated variable. Clearly, clauses consisting of a single literal force the assignment of these variables. Setting these variables may create new clauses of a single literal; set these variables and continue the process until all clauses have at least two literals or until a contradiction is reached, in which case no feasible assignment is possible. In the former case, setting the remaining variables to one satisfies all constraints, and there exists no feasible assignment with a greater weight of ones.

In the case that $\mathcal{F}$ is affine with width 2, we reduce the problem of finding a feasible solution to checking whether a graph is bipartite, and then use the bipartition to find the optimal solution. Notice that each constraint corresponds to a conjunction of constraints of the form $X_i = X_j$ or $X_i \neq X_j$. Create a vertex X_j for each variable X_j and for each constraint $X_i \neq X_j$, add an edge (X_i, X_j). For each constraint $X_i = X_j$, identify the vertices X_i and X_j; if this creates a self-loop, then clearly no feasible assignment is possible. Check whether the graph is bipartite; if not, then there is no feasible assignment. If so, then for each connected component of the graph choose the larger weight side of the bipartition, and set the corresponding variables to one. ■

5.2.2 Case 2: The APX-complete case

To prove Case 2, we must first show that the problems in this Case are in APX.

Lemma 5.9 If $\mathcal{F} \subset \mathcal{F}_4$, then the weighted MAX ONE($\mathcal{F}$) problem is in APX.

Proof: By Lemmas 5.4 and 5.2 it suffices to consider the unweighted case. In this case when all constraints are affine, then satisfying all constraints is essentially the problem of solving a linear system of equations over GF[2]. If the system is overdetermined, then no feasible solution exists. If the system is exactly determined, then the setting of all variables is forced, and we find the assignment with the maximum possible number of ones. If the system is underdetermined, then setting some number of variables arbitrarily determines the remainder of the solution; to be more precise, the variables $\vec{X}$ can be partitioned into $\vec{X'}$ and $\vec{X''}$ such that $\vec{X'} = A\vec{X''} \oplus b$ for some 0/1 matrix A and some 0/1 vector b (where matrix arithmetic is carried out over GF[2]). Setting the variables in $\vec{X''}$ to 1 with probability 1/2 thus ensures that the probability of each variable is 1 with probability 1/2. The expected number of ones is $n/2$, no worse than a factor of two from the maximum number. ■

We now must show that the problems in Case 2 are APX-hard. The basic idea of this proof is to show via implementations that all such problems can be reduced to MAX CUT. Recall the definitions of C-closed constraints and the constraints XOR and REP from Section 4. From here onwards all of our implementations will be perfect implementations. We begin with a preliminary lemma that we will need.

Lemma 5.10 Given a constraint f which is not 1-valid, the following hold:

1. If f is C-closed, then f perfectly implements REP and XOR.
2. If g is a constraint which is not C-closed, then $\{f, g\}$ implements an existential zero. If f is not 0-valid then $\{f, g\}$ also implements an existential one.
3. $\{f\}$ either implements an existential zero or perfectly implements the constraints REP and XOR.

Proof: If f is C-closed, then it is not 0-valid and we can use Lemma 4.14 to perfectly implement XOR and use XOR to implement REP (by using the constraints $X \oplus Z$, $Z \oplus Y$ for an auxiliary variable Z). This gives us Part (1) above.

For Part (2), if f is 0-valid, then the constraint $f(X_1, \ldots, X_k)$ implements an existential zero. Hence we can assume that f is neither 0-valid nor 1-valid. If f is not C-closed then Lemma 4.15 can be applied to perfectly implement the F and T constraints which are special cases of existential zero and one respectively. Lastly if f is C-closed, then by part (1) we can implement REP and XOR. We use these and g as follows: Let s be a satisfying assignment for g such that $\bar{s}$ does not satisfy s. W.l.o.g., assume that $s = 0^p 1^q$ and $p > 0$. Add the constraints $g(X_1, \ldots, X_p, Y_1, \ldots, Y_q)$

and the constraints $\mathrm{REP}(X_1, X_i)$ for all $i \in \{2, \ldots, p\}$ and the constraints $\mathrm{XOR}(X_1, Y_j)$ for all $j \in \{1, \ldots, q\}$. This forces X_1 to be assigned 0 and thus perfectly implements the F constraint. Using XOR we can now easily perfectly implement the T constraint.

Part (3) follows from the fact that if f is C-closed, then Part (1) applies, else we can apply Part (2) to $\{f, f\}$. ■

Definition 5.11 The constraint XOR_3 is defined to be the constraint $\mathrm{XOR}_3(X, Y, Z) = 1$ if and only if $X \oplus Y \oplus Z = 0$. The constraint XOR_4 is defined to be the constraint $\mathrm{XOR}_4(W, X, Y, Z) = 1$ if and only if $W \oplus X \oplus Y \oplus Z = 0$.

Lemma 5.12 The weighted problem MAX ONE($\{\mathrm{XOR}_3\}$) is APX-hard.

Proof: We reduce the MAX CUT problem to the weighted MAX ONE($\{\mathrm{XOR}_3\}$) problem as follows. Given a graph $G = (V, E)$ we create a variable X_v for every vertex $v \in V$ and a variable X_e for every edge $e \in E$. The weight w_v associated with the vertex variable X_v is 0. The weight w_e of an edge variable X_e is 1. For every edge e between u and v we create the constraint $X_e \oplus X_u \oplus X_v = 0$. It is clear that any 0/1 assignment to the X_v's define a cut and for an edge $e = \{u, v\}$, X_e is one iff u and v are on opposite sides of the cut. Thus solutions to the MAX ONE problem correspond to cuts in G with the objective function being the number of edges crossing the cut. ■

Lemma 5.13 The weighted problem MAX ONE($\{\mathrm{XOR}_4, \mathrm{XOR}\}$) is APX-hard.

Proof: The proof is similar to that of Lemma 5.12. In this case, given a graph $G = (V, E)$, we create the variables X_v for every $v \in V$, X_e for every $e \in E$ and one global variable Z (which is supposed to be zero) and $m \triangleq |E|$ auxiliary variables $y_1, \ldots, y_m$. For every edge $e = \{u, v\}$ in G we impose the constraints $X_e \oplus X_u \oplus X_v \oplus Z = 0$. In addition we throw in the constraints $Z \oplus y_i = 1$ for every $i \in \{1, \ldots, m\}$. Finally we make the weight of the vertex variables and Z zero and the weight of the edge variables and the auxiliary variables y_i is made 1. The optimum to this MAX ONE problem is MAX CUT$(G) + m$. Since MAX CUT$(G) \geq m/2$, we find that an α-approximation algorithm for MAX ONE($\{\mathrm{XOR}_4, \mathrm{XOR}\}$) yields an $\alpha' = \frac{\alpha}{3-2\alpha}$-approximation algorithm for MAX CUT. Notice that $\alpha' \to 1$ as $\alpha \to 1$. The APX-hardness of MAX CUT now implies the APX-hardness of MAX ONE($\{\mathrm{XOR}_4, \mathrm{XOR}\}$). ■

Lemma 5.14 Suppose $\{f\}$ implements the existential zero property and g is the constraint $(X_1 \oplus \cdots \oplus X_p = b)$ for some integer $p \geq 3$ and some $b \in \{0, 1\}$. Then the constraint set $\{f, g\}$ perfectly implements XOR_3.

Proof: Since $\{f\}$ perfectly implements the existential zero property, the set $\{f, g\}$ can perfectly implement $\{f, g\}|_0$ (using Lemma 5.7). In particular, $\{f, g\}$ can implement the constraints $X_1 \oplus X_2 = b$ and $X_1 \oplus X_2 \oplus X_3 = b$. Notice finally that the constraints $X_1 \oplus X_2 \oplus Y = b$ and $Y \oplus X_3 = b$ perfectly implement the constraint $X_1 \oplus X_2 \oplus X_3 = 0$. Thus $\{f, g\}$ perfectly implements the constraint XOR_3. ■

Lemma 5.15 Suppose $\{f\}$ perfectly implements the XOR constraint and g is the constraint $(X_1 \oplus \cdots \oplus X_p = b)$ for some integer $p \geq 3$ and some $b \in \{0, 1\}$. Then the constraint set $\{f, g\}$ either perfectly implements XOR_3 or XOR_4.

Proof: First observe that f can also implement the constraint REP since $X_1 \oplus Y = 1$ and $Y \oplus X_2 = 1$ implement the constraint $X_1 \oplus X_2 = 0$.

Next observe that we can assume w.l.o.g. that $b = 0$. If not the constraints $X_1 \oplus \cdots \oplus X_{p-1} \oplus Y = 1$ and $Y \oplus X_p = 1$ implement the constraint $X_1 \oplus \cdots \oplus X_p = 0$.

Now if p is odd, then the constraints $X_1 \oplus \cdots \oplus X_p = 0$ and REP(X_4, X_5), REP(X_6, X_7) and so on up to REP(X_{p-1}, X_p) implement the constraint $X_1 \oplus X_2 \oplus X_3 = 0$.

Lastly, if p is even, then the constraints $X_1 \oplus \cdots \oplus X_p = 0$ and REP(X_5, X_6), REP(X_7, X_8) and so on up to REP(X_{p-1}, X_p) implement the constraint $X_1 \oplus X_2 \oplus X_3 \oplus X_4 = 0$. ■

Theorem 5.16 *If $\mathcal{F} \subset \mathcal{F}_4$ and $\exists f_i \in \mathcal{F} - \mathcal{F}_i$ for every $i \in \{1, 2, 3\}$ then MAX ONE$(\mathcal{F})$ is APX-hard.*

Proof: Let $g(X_1, \ldots, X_q)$ be an affine constraint which not of width-2. As in the proof of Lemma 5.9 we use the fact that we can partition the variables of g into sets x' and x'' such that there exists a matrix A and vector b such that every assignment to x'' with the setting $x' = Ax'' \oplus b$, is a satisfying assignment to g. If every row of A has only one 1, then the constraint g has width-2. Hence there exists a row with at least two ones. By renaming the variables, this constraints can be expressed as saying $X_p = X_1 \oplus \cdots \oplus X_{p-1} \oplus 0/1$, where $p - 1 \geq 2$. Then the constraint $\exists X_{p+1}, \ldots, X_q$ s.t. $g(X_1, \ldots, X_q) = 1$ perfectly implements the constraint $X_1 \oplus X_2 \oplus \cdots \oplus X_p = 0/1$. Now let f be a non-1-valid constraint. If $\{f\}$ implements the existential zero property, then by Lemma 5.14 the set $\{f, g\}$ can implement XOR$_3$. Else, by Lemmas 5.10 and 5.15, $\{f\}$ perfectly implements XOR and $\{f, g\}$ either perfectly implements $\{\text{XOR}_3\}$ or $\{\text{XOR}_4\}$. Thus in any case $\{f, g\}$ can either perfectly implement the constraint XOR$_3$ or the set $\{\text{XOR}, \text{XOR}_4\}$, either of which is APX-hard. ■

5.2.3 Case 3: The poly-APX-Complete Case

We first show that the problems in this case are in poly-APX.

Lemma 5.17 *If $\mathcal{F} \subset \mathcal{F}_i$ for some $i \in \{1, 2, 3, 4, 5, 6, 7\}$ then MAX ONE$(\mathcal{F})$ can be approximated to within a factor of n.*

Proof: Schaefer's results imply a polynomial-time algorithm to compute a feasible solution. If the feasible solution has at least one 1, we are done. Else, iteratively try setting every variable to one and computing a feasible solution. Note that if $\mathcal{F}$ is affine (or 2CNF), then the constraints obtained by restricting some variable to be 1 remains affine (or resp. 2CNF), and thus this new class is still decidable. Lastly, a strongly 0-valid constraint set remains 0-valid after this restriction and is still decidable. If the decision procedure gives no non-zero solution, then the optimum is zero, else we output a solution of value at least 1. ■

We now turn to showing that this class of problems is poly-APX-hard. Our goal will be to perfectly implement the constraint $\bar{X}_1 + \cdots + \bar{X}_k$, for some $k \geq 2$. The following lemma shows that this will imply poly-APX-hardness.

Lemma 5.18 *If $f = \bar{X}_1 + \cdots + \bar{X}_k$, then MAX ONE$(\{f\})$ is poly-APX-hard.*

Proof: We do a reduction from MAX CLIQUE, which is known to be poly-APX-hard [1]. Given a graph G, construct a MAX ONE$(\{f\})$ instance consisting of a variable for every vertex in G and the constraint f is applied to every subset of k vertices in G which does not induce a clique. It may be verified that the optimum number of ones in any satisfying assignment to the instance created in this manner is $\max\{k-1, \omega(G)\}$, where $\omega(G)$ is the size of the largest clique in G. Given a solution to the MAX ONE$(\{f\})$ instance with $l \geq k$ ones, the set of vertices corresponding to the variables set to one form a clique of size l. If $l < k$, output any singleton vertex. Thus in all cases we obtain a clique of size at least $l/(k-1)$ vertices. Thus the existence of an α-approximation algorithm for the MAX ONE($\{f\}$) problem implies the existence of a $(k-1)\alpha$-approximation algorithm to the clique problem. The poly-APX-hardness of clique now implies the lemma. ■

The following lemma divides the remainder of the proof into two cases.

Lemma 5.19 If $\mathcal{F} \subset \mathcal{F}_i$ for some $i \in \{5, 6, 7\}$, but $\mathcal{F} \not\subset \mathcal{F}_j$ for any $j \in \{1, 2, 3, 4\}$, then either $\mathcal{F}$ perfectly implements every constraint in $\mathcal{F}|_{0,1}$ or $\mathcal{F}$ perfectly implements $\mathcal{F}|_0$ and every constraint in $\mathcal{F}$ is 0-valid.

To prove this lemma, we first need to show that there exists a non-C-closed constraint in $\mathcal{F}$.

Lemma 5.20 If $\mathcal{F} \subset \mathcal{F}_i$ for some $i \in \{5, 6, 7\}$, but $\mathcal{F} \not\subset \mathcal{F}_j$ for any $j \in \{1, 2, 3, 4\}$, then there exists a non-C-closed constraint in $\mathcal{F}$.

Proof: Notice that a C-closed 0-valid constraint is also 1-valid. A C-closed weakly positive constraint will also be weakly negative. Lastly a C-closed 2CNF constraint is a affine constraint of width 2. Thus if $\mathcal{F}$ is 0-valid, then the constraint which is not 1-valid is not C-closed. If $\mathcal{F}$ is weakly negative, the constraint which is not weakly positive is not C-closed. Similarly if $\mathcal{F}$ is 2CNF, then the constraint that is not affine is not C-closed. ■

Proof of Lemma 5.19: By Lemma 5.20 there exists a non C-closed constraint f in $\mathcal{F}$. Also since $\mathcal{F}$ is not 1-valid, there exists a constraint $g \in \mathcal{F}$ which is not 1-valid. By Lemma 5.10 the set $\{f, g\}$ can implement an existential zero and hence $\mathcal{F}$ can perfectly implement every constraint in the constraint set $\mathcal{F}|_0$. Furthermore, if $\mathcal{F}$ contains a non 0-valid constraint then it can perfectly implement $\mathcal{F}|_{0,1}$ (again by Lemma 5.10), else every element of $\mathcal{F}$ is 0-valid. ■

We now show that in the first case of Lemma 5.19, we can perfectly implement $\bar{X} + \bar{Y}$. We will then turn to the case in which all constraints are 0-valid and show that we can either perfectly implement $\bar{X}_1 + \cdots + \bar{X}_k$ or an existential one. If we can implement an existential one, then we are in the same situation as the first case. This will complete the proof of poly-APX-hardness.

Recall that we have a constraint in $\mathcal{F}$ that is not weakly positive and a constraint that is not affine.

<u>**Case I**</u> **:** $\mathcal{F}$ perfectly implements every constraint in $\mathcal{F}|_{0,1}$.

Lemma 5.21 If f is not weakly positive, then the constraint set $\{f\}|_{0,1}$ perfectly implements either XOR or $\bar{X} + \bar{Y}$.

Proof: Let $C = (\bar{X}_1 + \cdots + \bar{X}_p + Y_1 + \cdots + Y_q)$ be a maxterm in f with more than one negation i.e. $p \geq 2$. Substituting a 1 in place of variables $\bar{X}_3, \bar{X}_4, \ldots, \bar{X}_p$, a 0 in place of variables $Y_1, Y_2, \ldots, Y_q$, and existentially quantifying over all variables not in C, we get a constraint f' such that $(\bar{X}_1 + \bar{X}_2)$ is a maxterm in f'.

By definition of maxterm, f' must be satisfied whenever $X_1 \oplus X_2 = 1$. Now if f' is also satisfied when $X_1 = X_2 = 0$, we get the constraint $\bar{X}_1 + \bar{X}_2$, else we get the constraint $\mathrm{XOR}(X_1, X_2)$. ■

Lemma 5.22 The constraint set $\{\mathrm{XOR}\}$ can perfectly implement the constraint REP.

Proof: To perfectly implement $\mathrm{REP}(X, Y)$, we include the constraints $\mathrm{XOR}(X, Z)$ and $\mathrm{XOR}(Y, Z)$ for an auxiliary variable Z. ■

Lemma 5.23 Let g be a non-affine constraint. Then the constraint set $\{g, \mathrm{REP}, \mathrm{XOR}\}|_{0,1}$ can either perfectly implement the constraint $(X + \bar{Y})$ or $(\bar{X} + \bar{Y})$.

Proof: Since g is non-affine, we essentially have the following situation for three satisfying assignments s_1, s_2 and s_3 for g.

									$g()$
s_1	00...0	00...0	00...0	00...0	11...1	11...1	11...1	11...1	1
s_2	00...0	00...0	11...1	11...1	00...0	00...0	11...1	11...1	1
s_3	00...0	11...1	00...0	11...1	00...0	11...1	00...0	11...1	1
$s_1 \oplus s_2 \oplus s_3$	00...0	11...1	11...1	00...0	11...1	00...0	00...0	11...1	0
	00...0	$XX...X$	$YY...Y$	$ZZ...Z$	$\bar{Z}\bar{Z}...\bar{Z}$	$\bar{Y}\bar{Y}...\bar{Y}$	$\bar{X}\bar{X}...\bar{X}$	11...1	

Fixing the above variables to 0's and 1's as shown in the last row, and assigning replicated copies of three variables X, Y and Z (and their negations using XOR), we get a constraint $h(X, Y, Z)$ with the truth-table in Figure 1.

x \ yz	00	01	11	10
0	1	B	1	A
1	C	1	D	0

Figure 1: Truth-table of the constraint $h(X, Y, Z)$

The undetermined values in the table are indicated by the parameters A, B, C and D. The following analysis shows that for every possible value of these parameters, we can indeed perfectly implement an OR constraint using the constants 0 and 1.

$$\begin{aligned}
A = 0 &\implies \exists\, X\ h(Y, Z) = Y + \bar{Z} \\
A = 1, B = 0 &\implies h(0, Y, Z) = \bar{Y} + Z \\
A = 1, B = 1, C = 0 &\implies h(X, 0, Z) = X + \bar{Z} \\
A = 1, B = 1, C = 1, D = 0 &\implies h(X, Y, 1) = \bar{X} + \bar{Z} \\
A = 1, B = 1, C = 1, D = 1 &\implies h(1, Y, Z) = Y + \bar{Z}
\end{aligned}$$

■

Lemma 5.24 The constraint set $\{X + \bar{Y}, \mathrm{XOR}\}$ perfectly implements the constraint $\bar{X} + \bar{Y}$.

Proof: To perfectly implement $\bar{X} + \bar{Y}$, we create an auxiliary variable X'. We now add two constraints, namely $X' + \bar{Y}$, and $\mathrm{XOR}(X, X')$. Clearly, all constraints are satisfied only if $\bar{X} + \bar{Y}$ is satisfied. ■

Thus in all cases we are able to implement the constraint $\bar{X} + \bar{Y}$.

<u>**Case II**</u> : $\mathcal{F}$ can perfectly implement all constraints in $\mathcal{F}|_0$ and all constraints are 0-valid.

We now show that either we can perfectly implement $\bar{X} + \bar{Y}$, or perfectly implement a 1. If the former occurs, we are done, and if the latter, we can reduce to the previous case.

Lemma 5.25 If f is 0-valid and not weakly positive, then the constraint set $\{f\}|_0$ either perfectly implements $\bar{X}_1 + \cdots + \bar{X}_k$ for some $k \geq 2$ or it perfectly implements $X + \bar{Y}$ or REP.

Proof: Let $C = (\bar{X}_1 + \cdots + \bar{X}_p + Y_1 + \cdots + Y_q)$ be a maxterm in f with more than one negation i.e. $p \geq 2$ (such a maxterm exists since f is not weakly positive). Substituting a 0 in place of variables $Y_1, Y_2, \ldots, Y_q$, and existentially quantifying over all variables not in C, we get a constraint g such that $(\bar{X}_1 + \bar{X}_2 + \cdots + \bar{X}_p)$ is a maxterm in g. Consider an unsatisfying assignment s for g with the smallest number of 1's and let k denote the number of 1's in s; we know $k > 0$ since the original constraint 0-valid. WLOG assume that s assigns value 1 to the variables $X_1, X_2, \ldots, X_k$ and 0's to the remaining variables. It is easy to see that by fixing the variables $X_{k+1}, X_{k+2}, \ldots, X_p$ to 0, we get a constraint $g' = (\bar{X}_1 + \bar{X}_2 + \cdots + \bar{X}_k)$. If $k > 1$, then this perfectly implements the constraint $(\bar{X}_1 + \cdots + \bar{X}_k)$ and we are done.

Otherwise $k = 1$, i.e. there exists an unsatisfying assignment s which assigns value 1 to exactly one of the X_i's, say X_1. Now consider a satisfying assignment s' which assigns 1 to X_1 and has a minimum number of 1's among all assignments which assign 1 to X_1. The existence of such an assignment easily follows from C being a maxterm in g. WLOG assume that $s' = 1^i 0^{p-i}$. Thus the constraint g looks as follows:

	X_1	X_2	$X_3 \ldots X_i$	$X_{i+1} \ldots X_p$	$g()$
s_1	0	0	00...0	00...0	1
s_2	1	0	00...0	00...0	0
$s' = s_3$	1	1	11...1	00...0	1
s_4	0	1	__..._	00...0	?

Existential quantification over the variables $X_3, X_4, \ldots, X_i$ and fixing the variables X_{i+1} through X_p to 0 yields a constraint g' which is either $(X_1 + \bar{X}_2)$ or $\mathrm{REP}(X_1, X_2)$. The lemma follows. ■

If we can perfectly implement $X + \bar{Y}$, then the following lemma shows that we can essentially perfectly implement a 1, and thus we can reduce to Case I. We use the constraint function $T(X_i) = X_i$ to represent constraints $X_i = 1$.

Lemma 5.26 If MAX ONE$(\mathcal{F} \cup \{X + \bar{Y}\})$ is α-approximable for some function α, then so is MAX ONE$(\mathcal{F} \cup \{T\})$.

Proof: Given an instance $\mathcal{I}$ of MAX ONE$(\mathcal{F} \cup \{T\})$ we construct an instance $\mathcal{I}'$ of MAX ONE$(\mathcal{F} \cup \{X + \bar{Y}\})$ as follows. The variable set of $\mathcal{I}'$ is the same as that of $\mathcal{I}$. Every constraint from $\mathcal{F}$ in $\mathcal{I}$ is also included in $\mathcal{I}'$. The only remaining constraints are of the form $X_i = 1$ for some variables X_i (imposed by the constraint T). We simulate this constraint in $\mathcal{I}'$ with $n - 1$ constraints of the form $X_i + \bar{X}_j$ for every $j \in \{1, \ldots, n\}$, $j \neq i$. Every non-zero solution to the resulting instance $\mathcal{I}'$ is also a solution to $\mathcal{I}$, since the solution must have $X_i = 1$ or else every $X_j = 0$. Thus the resulting instance of MAX ONE$(\mathcal{F} \cup \{X + \bar{Y}\})$ has the same objective function and the same feasible space and is hence at least as hard as the original problem. ■

Now by Lemma 5.25 the only remaining subcase is if we can perfectly implement REP. The following lemma shows that in this case we can either perfectly implement $\bar{X} + \bar{Y}$ or $X + \bar{Y}$. If we can do the former, we are done, and if the latter, we can use $X + \bar{Y}$ to perfectly implement the T constraint, and reduce to the previous case. Hence in either case we are finished.

Lemma 5.27 If f is 0-valid constraint and non-affine, then MAX ONE$(\{f, \mathrm{REP}\})$ perfectly implements either the constraint $(\bar{X} + \bar{Y})$ or the constraint $(X + \bar{Y})$.

Proof: Schaefer [15] shows that if f be a non-affine constraint, then there exist two satisfying assignments s_1 and s_2 such that $s_1 \oplus s_2$ is not a satisfying assignment for f. Using this fact and the fact that f is 0-valid, we essentially have the following situation:

					$g()$
	00...0	00...0	00...0	00...0	1
s_1	00...0	00...0	11...1	11...1	1
s_2	00...0	11...1	00...0	11...1	1
$s_1 \oplus s_2$	00...0	11...1	11...1	00...0	0
	00...0	$XX...X$	$YY...Y$	$ZZ...Z$	

Fixing the above variables to 0's as shown in the last row, and assigning replicated copies of three variables X, Y and Z, we get a constraint $h(X, Y, Z)$ with the truth-table in Figure 1. The lemma now follows using an analysis identical to the one used in Lemma 5.23. ■

5.3 Case 4: Finding a Solution of Positive Value is NP-Hard

The following two lemmas prove Case 4 of Theorem 2.3.

Lemma 5.28 If $\mathcal{F} \subset \mathcal{F}_8$, then $\mathrm{SAT}(\mathcal{F})$ is in P.

Proof: Follows trivially since every instance is satisfiable. The assignment of 0's to every variable is a satisfying assignment to every instance. ■

Lemma 5.29 If $\mathcal{F} \not\subset \mathcal{F}_j$, for any $j \in \{1, \ldots, 7\}$, then the problem of finding solutions of non-zero value to a given instance of (unweighted) MAX ONE($\mathcal{F}$) is NP-hard.

Proof: Given a constraint $f : \{0,1\}^k \to \{0,1\}$ and an index $i \in [k]$, let $f|_i$ be the constraint mapping $\{0,1\}^{k-1}$ to $\{0,1\}$ given by

$$f|_i(X_1, \ldots, X_k) \stackrel{\text{def}}{=} f(X_1, \ldots, X_{i-1}, 1, X_{i+1}, \ldots, X_k) \wedge f(X_1, \ldots, X_{i-1}, 0, X_{i+1}, \ldots, X_k).$$

Let $\mathcal{F}'$ be the set of constraints defined as follows:

$$\mathcal{F}' \stackrel{\text{def}}{=} \mathcal{F} \cup \{f|_i \quad | \quad f \in \mathcal{F}, i \in [k]\}.$$

I.e., $\mathcal{F}'$ is the set of constraints obtained by setting none or one of the variables of $\mathcal{F}$ to 1. We will argue that deciding $\mathrm{SAT}(\mathcal{F}')$ is NP-hard and then that deciding $\mathrm{SAT}(\mathcal{F}')$ reduces to finding non-zero solutions to MAX ONE($\mathcal{F}$).

First observe that $\mathcal{F}' \not\subset \mathcal{F}_j$, for $j \in \{1, \ldots, 8\}$. In particular it is not 0-valid, since $\mathcal{F}$ is not strongly 0-valid. Hence, once again applying Schaefer's result, we find that deciding $\mathrm{SAT}(\mathcal{F}')$ is NP-hard.

Now given an instance of $\mathrm{SAT}(\mathcal{F}')$ on variables $V_1, \ldots, V_n$ with constraints $C_1, \ldots, C_m$, with $C_1, \ldots, C_{m'} \in \mathcal{F}$ and $C_{m'+1}, \ldots, C_m \in \mathcal{F}' \setminus \mathcal{F}$, consider the instance of MAX ONE($\mathcal{F}$) defined on variable set

$$W_1, \ldots, W_{k+1}, Y_1, \ldots, Y_n, Z_1, \ldots, Z_n$$

with the following constraints:

1. Let f be a non-1-valid constraint in $\mathcal{F}$. We introduce the constraint $f(W_1, \ldots, W_k)$.

2. For every constraint $C_i(V_{i_1}, \ldots, V_{i_k})$, $1 \le i \le m'$, we introduce two constraints $C_i(Y_{i_1}, \ldots, Y_{i_k})$ and $C_i(Z_{i_1}, \ldots, Z_{i_k})$.

3. For every constraint $C_i(V_{i_1}, \ldots, V_{i_{k-1}})$, $m'+1 \le i \le m$, we introduce $2(n+k+1)$ constraints. For simplicity of notation, let $C_i(V_{i_1}, \ldots, V_{i_{k-1}}) == g(1, V_{i_1}, \ldots, V_{i_{k-1}}) \wedge g(0, V_{i_1}, \ldots, V_{i_{k-1}})$ where $g \in \mathcal{F}$. The $2(n+k+1)$ constraints are:

- $g(W_j, Y_{i_1}, \ldots, Y_{i_{k-1}})$, for $1 \le j \le k+1$.
- $g(Z_j, Y_{i_1}, \ldots, Y_{i_{k-1}})$, for $1 \le j \le n$.
- $g(W_j, Z_{i_1}, \ldots, Z_{i_{k-1}})$, for $1 \le j \le k+1$.
- $g(Y_j, Z_{i_1}, \ldots, Z_{i_{k-1}})$, for $1 \le j \le n$.

We now show that the instance of MAX ONE($\mathcal{F}$) created above has a non-zero satisfying assignment if and only if the instance of SAT($\mathcal{F}'$) has a satisfying assignment. Let $s = s_1 s_2 \ldots s_k$ be a satisfying assignment for the non 1-valid constraint f chosen above. First if $V_1, \ldots, V_n$ form a satisfying assignment to the instance of SAT($\mathcal{F}'$), then we claim that the assignment $W_j = s_j$ for $1 \le j \le k$, $W_{k+1} = 1$ and $Y_j = Z_j = V_j$ for $1 \le j \le n$ is a satisfying assignment to the instance of MAX ONE($\mathcal{F}$) which has at least one 1 (namely W_{k+1}). Conversely, let some non-zero setting $W_1, \ldots, W_{k+1}, Y_1, \ldots, Y_n, Z_1, \ldots, Z_n$ satisfy the instance of MAX ONE($\mathcal{F}$). W.l.o.g./ assume that one of the variable $W_1, \ldots, W_{k+1}, Y_1, \ldots, Y_n$ is a 1. Then we claim that the setting $V_j = Z_j, 1 \le j \le n$ satisfies the instance of SAT($\mathcal{F}'$). It is easy to see that the constraints $C_i(V_{i_1}, \ldots, V_{i_k}), 1 \le i \le m'$, are satisfied. Now consider a constraint $C_i(V_{i_1}, \ldots, V_{i_{k-1}}) = g(0, V_{i_1}, \ldots, V_{i_{k-1}}) \wedge g(1, V_{i_1}, \ldots, V_{i_{k-1}})$. Since at least one of the variables in the set $W_1, \ldots, W_k$ is a 0 and at least one of the variables in the set $W_1, \ldots, W_{k+1}, Y_1, \ldots, Y_n$ is 1, we know that both $g(0, Z_{i_1}, \ldots, Z_{i_{k-1}})$ and $g(1, Z_{i_1}, \ldots, Z_{i_{k-1}})$ are satisfied and hence $C_i(V_{i_1}, \ldots, V_{i_{k-1}}) = 1$. Thus the reduced instance of MAX ONE($\mathcal{F}$) has a non-zero satisfying assignment if and only if the instance of SAT($\mathcal{F}'$) is satisfiable. ∎

5.4 Case 5: Finding Any feasible Solution is NP-Hard

The following lemma proves Case 5 of Theorem 2.3.

Lemma 5.30 If, for all $j \in \{1, \ldots, 8\}$, $\mathcal{F} \not\subseteq \mathcal{F}_j$, then deciding SAT($\mathcal{F}$) is NP-hard.

Proof: Follows from Schaefer's theorem. ∎

Acknowledgments

Many thanks to Nadia Creignou, Oded Goldreich, Greg Sorkin and Luca Trevisan for their valuable comments.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the intractability of approximation problems. *Proceedings of the* 33rd *Symposium on Foundations of Computer Science*, IEEE, 1992.

[2] T. Asano, T. Ono, and T. Hirata. Approximation algorithms for the maximum satisfiability problem. *Scandanavian Workshop on Algorithmic Theory* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1097, ed., Springer-Verlag, 1996.

[3] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP and non-approximability — towards tight results. (Version 3). *ECCC Technical Report* number TR95-024, 1995.

[4] N. Creignou. A Dichotomy Theorem for Maximum Generalized Satisfiability Problems. *Journal of Computer and System Sciences*, 51:3, pp. 511–522, 1995.

[5] T. Feder and M. Vardi. Monotone monadic SNP and constraint satisfaction. *Proceedings of the* 25th *Annual Symposium on Theory of Computing*, ACM, 1993.

[6] M. Goemans and D. Williamson. New 3/4-approximation algorithms for MAX SAT. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994.

[7] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[8] S. Khanna and R. Motwani. Towards a Syntactic Characterization of PTAS. *Proceedings of the* 28th *Annual Symposium on Theory of Computing*, ACM, 1996.

[9] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On Syntactic versus Computational Views of Approximation. *Proceedings of the* 35th *Symposium on Foundations of Computer Science*, IEEE, 1994, pp. 819–830.

[10] S. Khanna and M. Sudan. The Optimization Complexity of Constraint Satisfaction Problems. *Electonic Colloquium on Computational Complexity*, Technical report no. TR96-028, 1996.

[11] S. Khanna, M. Sudan and L. Trevisan. In preparation.

[12] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:1, pp. 155–171, 1975.

[13] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43, pp. 425–440, 1991.

[14] E. Petrank. The Hardness of Approximation: Gap Location. *Computational Complexity*, v. 4, 1994.

[15] T. Schaefer. The complexity of satisfiability problems *Proceedings of the* 10th *Annual Symposium on Theory of Computing*, ACM, 1978.

[16] L. Trevisan, G. Sorkin, M. Sudan and D. Williamson. Gadgets, approximation and linear programming. *Proceedings of the* 37th *Symposium on Foundations of Computer Science*, IEEE, 1996.

[17] M. Yannakakis, On the approximation of maximum satisfiability. *Journal of Algorithms*, vol. 17, pages 475–502, 1994.