The Optimization Complexity of Constraint Satisfaction Problems

Sanjeev Khanna* Madhu Sudan†

Abstract

In 1978, Schaefer [12] considered a subclass of languages in NP and proved a "dichotomy theorem" for this class. The subclass considered were problems expressible as "constraint satisfaction problems", and the "dichotomy theorem" showed that every language in this class is either in P, or is NP-hard. This result is in sharp contrast to a result of Ladner [9], which shows that such a dichotomy does not hold for NP, unless NP=P.

We consider optimization version of the dichotomy question and show an analog of Schaefer's result for this case. More specifically, we consider optimization version of "constraint satisfaction problems" and show that every optimization problem in this class is either solvable *exactly* in P, or is MAX SNP-hard, and hence not *approximable* to within some constant factor in polynomial time, unless NP=P. This result does not follow directly from Schaefer's result. In particular, the set of problems that turn out to be hard in this case, is quite different from the set of languages which are shown hard by Schaefer's result.

*`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.

1 Introduction

The deepening connection between the existence of probabilistically checkable proofs and optimization problems along with efforts to unify the syntactic and computational views of approximation, has resulted in a significant improvement in our understanding of the approximability of optimization problems and in identifying the structure of canonical hard problems. The recent chain of work has both resolved the approximability of a large number of important combinatorial problems and has helped identify natural complete problems for the approximation classes. At this juncture it seems reasonable to ask if we can characterize which optimization problems are easy and which ones hard; and to ask if there are any characteristic features of hard problems that can be isolated.

Of course, no complete characterization is possible: Rice's theorem allows one to disguise an optimization problem so cleverly that it would be undecidable to determine if a given problem is NP-hard or polynomial time solvable. Even if the problem is presented in its simplest form — it may be the case that the answer need not be "easy" or "NP-hard" — this is established by a theorem of Ladner [9].

In the presence of such barriers one is forced to weaken one's goals and focus one's attention onto a restricted subclass of NPO (optimization problems within NP) in the hope that some features of hard problems can be isolated from this subclass. Our choice of the appropriate restriction comes from the work of Schaefer [12] who carried out an analogous investigation in the case of decision problems. Schaefer considered a restriction of NP that he called "satisfiability problems" and successfully characterized every problem in this (infinite) class as being easy (polynomial time decidable) or hard (NP-hard). He refers to this as a "dichotomy theorem", since it partitions the class of problems studied into two polynomial time equivalent classes. A further study along these lines — looking for other dichotomic classes within NP — was carried out more recently by Feder and Vardi [5]. They suggest several promising classes which may show such a dichotomy.

The works of Schaefer and Feder and Vardi motivates our class of optimization problems which we refer to as "constraint satisfaction problems"*. These are problems obtained from Schaefer's class of decision problems in a natural fashion and are a subset of the problems in the class MAX SNP. Recall that MAX SNP is the class of optimization problems defined by Papadimitriou and Yannakakis [10] based on syntactic prescriptions. All problems in this class are known to be approximable to within some constant factor, and the complete problems in the class are known to be non-approximable in polynomial time to within some constant factor, unless NP=P [3].

In what follows we describe this class formally and present our main result.

1.1 Definitions and Main Result

We start with the definition of a constraint.

Definition 1 [Constraint] *A* constraint *is a function* $f : \{0,1\}^k \rightarrow \{0,1\}$. *We say* f *is* satisfied *by an assignment* $s \in \{0,1\}^k$ *if* $f(s) = 1$. *We refer to* k *as the* arity *of the constraint* f. *A constraint with no satisfying assignments is called* unsatisfiable.

Often we apply a constraint f of arity k to a subset of k variables from a larger set. In such cases we think of f as a constraint on the larger set of variables.

Definition 2 [Constraint Application] *Given* n *boolean variables* $X_1, \ldots, X_n$ *and a constraint* f *of arity* k*, and indices* $i_1, \ldots, i_k \in \{1, \ldots, n\}$*, the pair* $(f, (i_1, \ldots, i_k))$ *is referred to as an* application *of the constraint* f *to* $X_1, \ldots, X_n$. *An assignment* $X_i = s_i$ *for* $i \in \{1, \ldots, n\}$ *and* $s_i \in \{0,1\}$ satisfies *the application if* $f(s_{i_1}, \ldots, s_{i_k}) = 1$.

Definition 3 [Constraint Set] *A* constraint set $\mathcal{F} = \{f_1, \ldots, f_l\}$ *is a finite collection of constraints.*

Definition 4 [Constraint Satisfaction Problem (CSP($\mathcal{F}$))] *Given a constraint set* $\mathcal{F}$*, the* constraint satisfaction problem CSP($\mathcal{F}$) *is defined as follows :*

*In the terminology of Feder and Vardi, these should really be referred to as constraint satisfaction problems over a boolean domain. We drop the suffix in the interest of readability.

INPUT : *A collection of* m *constraint applications of the form* $\{(f_j, (i_1(j), \ldots, i_{k_j(j)}))\}_{j=1}^m$, *on boolean variables* $X_1, X_2, ..., X_n$ *where* $f_j \in \mathcal{F}$ *and* k_j *is the arity of* f_j.

OBJECTIVE : *Find a boolean assignment to* X_i*'s so as to maximize the number of applications of the constraints* $f_1, \ldots, f_m$ *that are satisfied by the assignment.*

Notice that the above definition gives a new optimization problem for every family $\mathcal{F}$. Schaefer's class of decision problems SAT($\mathcal{F}$) can be described in terms of the above as: The members of SAT($\mathcal{F}$) are all the instances of CSP($\mathcal{F}$) whose optimum equals the number of applied constraints (i.e., all constraints are satisfiable). Schaefer's dichotomy theorem essentially shows that the only families $\mathcal{F}$ for which SAT($\mathcal{F}$) is in P, is if all constraints in $\mathcal{F}$ are either satisfied by the all zeroes assignment, the all ones assignment or all constraints are linear constraints over GF(2) or all constraints are Horn clauses.

Our main result asserts that a dichotomy holds for CSP($\mathcal{F}$) as well. However the characterization of the easy functions and the hard ones is quite different. (In particular, many more constraint sets $\mathcal{F}$ are hard now.) In order to describe our result fully we need some more definitions.

Definition 5 *Given a constraint set* $\mathcal{F}$*, the constraint set* $\mathcal{F}'$ *is the set of constraints* f *in* $\mathcal{F}$ *which are satisfiable.*

It is easy to see that for any constraint set $\mathcal{F}$, an instance of CSP($\mathcal{F}$) can be mapped to an instance of CSP($\mathcal{F}'$), such that the objective function value is preserved on each input assignment. Hence our characterizations will essentially be characterizations of CSP($\mathcal{F}'$).

Definition 6 [i-valid function] *For* $i \in \{0, 1\}$*, a function* f *of arity* k *is called* i-valid *if it is satisfied by the assignment* i^k.

Definition 7 [Minterm] *Given a function* f *on variable* $x_1, \ldots, x_k$*, a collection of literals* $x_{i_1}, \ldots, x_{i_l}, \overline{x_{j_1}}, \ldots, \overline{x_{j_m}}$ *is called a minterm of* f *if it satisfies the following properties:*

1. *Any assignment* $s = s_1, \ldots, s_k$*, which satisfies* $s_{i_1} = \cdots = s_{i_l} = 1$ *and* $s_{j_1} = \cdots = s_{j_m} = 0$ *satisfies* f.
2. *The collection is minimal with respect to property (1).*

Definition 8 [Positive and Negative Minterms] *A minterm of* f *which consists only of unnegated variables is called a positive minterm. A minterm which consists only of negated variables is called a negative minterm.*

Definition 9 [2-Monotone Function] *A function* f *is called* 2-monotone *if it has at most two minterms such that at most one of them is positive and at most one is negative.*

The main result of this paper is as stated below.

Theorem 1 *The problem* CSP($\mathcal{F}$) *is always either in* P *or is* MAX SNP*-hard. Furthermore, it is in* P *if and only if one of the following conditions is true:*

1. *Every* $f \in \mathcal{F}'$ *is 0-valid.*
2. *Every* $f \in \mathcal{F}'$ *is 1-valid.*
3. *Every* $f \in \mathcal{F}'$ *is 2-monotone.*

This theorem follows from Lemmas 1,2,3 and 14. A second result that follows easily as a consequence of Schaefer's result and our notion of approximation preserving reductions is the following:

Theorem 2 *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.*

Schaefer's result characterizes which function families are easy to decide and which ones are hard; the result is described in the Appendix A for sake of completeness.

Discussion The main feature of Theorem 1 is that the family of constraint sets which lead to hard problems is significantly larger than in Schaefer's case. This is not surprising given that problems such as 2SAT and Linear systems over GF(2) are easy problems for the decision version and the maximization versions are known to be hard, even to approximate [10, 2]. Nevertheless, the set of problems that are shown to be easy is extremely small. CSP($\mathcal{F}$) for $\mathcal{F}$ which is 0-valid or 1-valid is really a trivial problem; leaving only the class of 2-monotone functions as somewhat interesting. But the class of functions with such properties seems to be really small and we know of no natural optimization problems which have this property. Thus, we feel, that the correct way to interpret Theorem 1 is to think of it as saying that *almost* every constraint satisfaction problem is hard (of course, we don't mean that formally), and that a result showing that an optimization problem can be expressed as a constraint satisfaction problem should essentially be viewed as hinting at a negative result (while on the surface this would actually be a positive result).

Another interesting feature of the above result is that the dichotomy holds for two different properties — the complexity and the approximability — simultaneously. *Easy* problems are easy to compute *exactly* and the hard ones are *hard* to even *approximate*. The middle regime — problems that are easy to approximate but hard to compute exactly — are ruled out. This may be somewhat surprising initially, but becomes inevitable once the form of approximation preserving reductions we use here becomes clear. Essentially all reductions we use are exactly those that might be used for *exact* optimization problems. In fact the ease with which these reductions apply is the reason why Theorem 2 falls out easily from this paper.

The technical aspects of the proof of the dichotomy theorem may be of some independent interest. In order to prove such a theorem, one needs to find succinct characterizations of what makes a function, say 2-monotone, as well as a succinct proof when a function is *not* 2-monotone. We find such a characterization, in Lemma 8, which turns out to be useful in establishing Theorem 1.

One technical nit-picky point that we face in this study is the role of constants and repetitions in CSP. In particular, if $f \in \mathcal{F}$, should $f|_{x_1=0}$ given by $f|_{x_1=0}(x_2, \ldots, x_k) = f(0, x_2, \ldots, x_k)$ also be considered a member of the constraint set? Similarly should $f'(x_1, x_2) = f(x_1, \ldots, x_1, x_2, \ldots, x_2)$ be considered a member of the constraint set. Allowing for these repetitions and constants makes the analysis much easier; however they may change the complexion of the problems significantly. For instance given a set of linear equalities of the form $\sum_i x_i = 0$, it is trivial to find an assignment which satisfies all the equations — namely the all 0's assignment. However once one is allowed to fix some variables to the constant 1, the problem no longer remains easy. In this paper initially we assume we can use constants and repetitions to make our analysis simpler. Later we remove the assumptions — and Theorem 1 and Theorem 2 is shown without the use of any constants or repetition. In fact, in the process we remove a minor irritant from Schaefer's proof which actually needed to use repetitions.

Other Work It has recently been brought to our attention that Theorem 1 has been independently discovered by Creignou [4].

Rest of this paper Sections 2 and 3 are devoted to proving Theorem 1. In Section 4 we show how to prove Theorem 2. Finally, in Section 5, we show how to eliminate the replication assumption from Schaefer's proof.

2 Polynomial Time Solvability

From this section onwards we omit the notation $\mathcal{F}'$ and assume we have a constraint $\mathcal{F}$ such that $\mathcal{F} = \mathcal{F}'$.

Lemma 1 *The problem* 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 2 *The problem* 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 3 *The problem* CSP($\mathcal{F}$) *is in* P *if each* f_i *is a 2-monotone function.*

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. Now we proceed as follows for each of the above classes of constraints :

- For a constraint C of the form (a), create a new node e_C and add an edge from each x_i to e_C of cost ∞ and a unit cost edge from e_C to T.
- For a constraint C 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 unit cost.
- Finally, for a constraint C 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 unit cost edges from F and to T by a unit cost edge from e_C to $\overline{e_C}$.

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 number of constraints that can be left unsatisfied. ■

3 Proof of MAX SNP-Hardness

In this section we prove that a constraint set which is not entirely 0-valid or entirely 1-valid or entirely 2-monotone gives a MAX SNP-hard problem. The main MAX SNP-hard problem which we reduce to any of these new ones is the MAX CUT problem shown to be MAX SNP hard by Papadimitriou and Yannakakis [10]. Initially 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 function is not 2-monotone, then it provides a MAX SNP hard problem. We then use the availability of functions that are not 0-valid or 1-valid to 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 the hardness lemma. We first start with some notation.

3.1 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 $O(s) = \{i \mid s_i = 1\}$. The notation $s[0 \to *]$ denotes the set of all assignments s' such that $O(s) \subseteq O(s')$ and similarly, $s[1 \to *]$ denotes the set of all assignments s' such that $Z(s) \subseteq Z(s')$.

Definition 10 [Unary Functions] *The functions* $T(X) = X$ *and* $F(X) = \bar{X}$ *are called* unary functions.

Definition 11 [XOR and REP Functions] *The function* $f(X, Y) = X \oplus Y$ *is called the* XOR *function and its complement function, namely* $X = Y$*, is called the* REP *function.*

Definition 12 [C-closed Function] *A function* f *is called* C-closed (or complementation-closed) *if for all assignments* s*,* $f(s) = f(\bar{s})$.

Definition 13 [v-Consistent Set] *A set* V *of positions is* v-consistent *for a constraint* f *iff every assignment with all variables occupying the positions in* V *set to value* v *is a satisfying assignment for* f.

3.2 α-Implementations and MAX SNP-Hard Functions

We next describe the primary form of a reduction which we use to give the hardness results. As pointed out by Papadimitriou and Yannakakis, in order to get hardness of approximation results, the reductions used need to satisfy certain approximation preserving features. Here we show how to implement a given function f using a family of other functions $\mathcal{F}$, so as to be useful in approximation preserving reductions.

Definition 14 [α-Implementation] *An instance of* CSP($\mathcal{F}$) *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 boolean function* $f(\vec{X})$*, where* α *is a positive integer, iff the following conditions are satisfied:*

(a) *no assignment of values to* $\vec{X}$ *and* $\vec{Y}$ *can satisfy more than* α *constraints,*

(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, and finally*

(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.*

We refer to the set $\vec{X}$ *as the* function variables *and the set* $\vec{Y}$ *as the* auxiliary variables.

Thus a function f 1-implements itself. We will say that CSP($\mathcal{F}$) implements function f if it α_f-implements f for some constant α_f. The following lemma shows that the α-implementations of functions compose together. The criteria for "implementation" given above are somewhat more stringent than used normally. While properties (1)-(3) are perhaps seen elsewhere, property (4) is somewhat more strict, but turns out to be critical in composing implementations together.

Lemma 4 [Composition Lemma] *If* CSP($\mathcal{F}_f$) *can* α_f*-implement a function* f*, and* CSP($\mathcal{F}_g$) *can* α_g*-implement a function* $g \in \mathcal{F}_f$*, then* CSP($\{\mathcal{F}_f \setminus \{g\}) \cup \mathcal{F}_g$) *can* α*-implement the function* f *for some constant* α.

Proof: Let β be the number of occurrences of a constraint involving the function g in the CSP($\mathcal{F}_f$) instance α_1-implementing f, then clearly, by replacing each occurrence of g by its α_2-implementation, we obtain a CSP($\{\mathcal{F}_f \setminus \{g\}) \cup \mathcal{F}_f$) instance which α-implements f for $\alpha = \alpha_1 + \beta(\alpha_2 - 1)$. ∎

The MAX SNP-hardness of MAX CUT implies that CSP($\{$XOR$\}$) is MAX SNP-hard and hence the below :

Lemma 5 *If* CSP($\mathcal{F}$) *can implement the* XOR *function, then* CSP($\mathcal{F}$) *is* MAX SNP-*hard.*

Lemma 6 CSP($\{f, T, F\}$) *can implement the* XOR *function if* f *is either the function* $X + Y$ *or* $X\bar{Y}$ *or* $\bar{X} + \bar{Y}$.

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

Lemma 7 CSP($\{f, T, F\}$) *can implement the* REP *function if* f *is the function* $\bar{X} + Y$.

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

3.3 Characterizing 2-Monotone Functions

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, which turns out to be useful.

Lemma 8 [Characterization Lemma] *A function f is a 2-monotone function if and only if all the following conditions are satisfied:*
(a) for every satisfying assignment s of f, either $s[1 \to *]$ or $s[0 \to *]$ is a set of satisfying assignments,
(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 function can be expressed in DNF form as the sum of its minterms. For a 2-monotone function 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 function has at most one positive and one negative minterm.

Conversely, if a function is not 2-monotone, then it either has a minterm which is not monotone positive or negative or it has more than one positive (or negative) minterm. In the former case, the function will violate Property (a), and in the latter one of Properties (b) or (c). ■

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

3.4 MAX SNP-hardness of Non 2-Monotone Functions

We now use the characterization from the previous subsection 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 9 [Hardness Lemma] *For any function f which is not 2-monotone,* CSP($\{f, T, F, \text{REP}\}$) *can implement the function* XOR.

The proof of this lemma is non-trivial and appears in Appendix B due to space limitations.

3.5 Implementing the REP Function

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 functions which are not 0-valid and not 1-valid. It turns out that the case in which the same function 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 10 [Replication Lemma] *Let f be a non-trivial function which is C-closed and is neither 0-valid nor 1-valid. Then an instance of* CSP($\{f\}$) *can implement the* REP *function.*

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 creating an instance $\mathcal{I}$ of CSP($\{f\}$) as follows. For each satisfying assignment s, there are $\binom{2k}{i}\binom{2k}{k-i}$ constraints in $\mathcal{I}$ such that every i-variable subset of S_X appears in place of 0's in S_X 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.

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 $\mathcal{I}$. 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 is hence 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 unsatisfied therefore. 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 unsatisfied therefore. 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 instance $\mathcal{I}$ of CSP($\{f\}$) 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 :

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

Consider the constraints $f(X, X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_q)$ and $f(Y, X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_q)$. 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 $f_v(Y, X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_q)$ is by assigning Y an identical value. Thus these set of constraints α-implements the function $X = Y$ where α is simply the total number of constraints; all constraints can be satisfied iff $X = Y$ and otherwise, there exists an assignment to variables in S_X and S_Y such that precisely $\alpha - 1$ constraints are satisfied.

■

3.6 Implementing the Unary Functions

If the function(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 11 [Unary Lemma] *Let f_0 and f_1 be two non-trivial functions, possibly identical, which are not 0-valid and 1-valid respectively. Then if neither f_0 nor f_1 is C-closed, an instance of* CSP($\{f_0, f_1\}$) *can implement both the unary functions $T(.)$ and $F(.)$.*

Proof: We will only sketch the implementation of function $T(.)$; the analysis for the function $F(.)$ is identical. Now suppose neither f_v is C-closed, $v \in \{0, 1\}$. We begin by considering an instance each of CSP($\{f_0\}$) and CSP($\{f_1\}$), say $\mathcal{I}_0$ and $\mathcal{I}_1$ respectively. Both of these instances are constructed in a manner identical to the instance $\mathcal{I}_A$ above. 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 functions 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 - 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 function 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 function $T(.)$. The function $F(.)$ can be implemented in an analogous manner.

■

3.7 REP **Helps Implement** MAX SNP**-Hard Functions**

Lemma 12 *Suppose f is a non-trivial function which is neither 0-valid nor 1-valid. Then* CSP($\{f, \text{REP}\}$) *implements the* XOR *function.*

Proof: Without loss of generality, assume $s = 0^p1^q$ is a satisfying assignment for f. We consider two disjoint set of variables $S_X = \{X_1, X_2, ..., X_p\}$ and $S_Y = \{Y_1, Y_2...., Y_q\}$. Consider the CSP($\{f, \text{REP}\}$) instance which consists of constraints REP(X_1, X_i) for $i \in [2..p]$, constraints REP(Y_1, Y_j) for $j \in [2..q]$ and the constraint $f(X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_q)$. It is now easy to verify that this yields a $(p + q - 1)$-implementation of the function $X_1 \oplus Y_1$ if $\bar{s} = 1^p0^q$ is a satisfying assignment, and of the function $\bar{X}_1Y_1$ otherwise. Now an application of the Composition Lemma yields the lemma. ■

Corollary 1 *Suppose f is a non-trivial function which is neither 0-valid nor 1-valid. Then* CSP($\{f, \text{REP}\}$) *is* MAX SNP*-hard.*

Proof: Immediately follows from Lemma 5 and Lemma 12 above. ■

3.8 Unary Functions Help Implement either REP or MAX SNP-Hard Functions

Lemma 13 *Let f be a function which is not 2-monotone. Then* CSP($\{f, T, F\}$) *can implement either the* XOR *or the* REP *function.*

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 function 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 later case can't arise since it would imply that f is a single variable function. 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^{th} and the j^{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 $f(X_1, X_2, Y_1, Y_2, ..., Y_{k-2})$ and constraints of the form $T(Y_i)$ for each Y_i appearing in $O(s) \cap O(s_1) \cap O(s_2)$ and of the form $F(Y_i)$ for each Y_i appearing in $Z(s) \cap Z(s_1) \cap Z(s_2)$. It is now easy to verify that this set of constraints implements one of the functions $X_1 + X_2$, $X_1 \oplus X_2$, $\bar{X}_1 + \bar{X}_2$, $\bar{X}_1 + X_2$ or $\overline{X_1 \oplus X_2}$. The former three implement $X_1 \oplus X_2$ while the later two implement the constraint $X_1 = X_2$. ■

The following is a straightforward corollary.

Corollary 2 *Let f be a function which is not 2-monotone. Then* CSP($\{f, T, F\}$) *is* MAX SNP*-hard.*

Proof: If CSP($\{f, T, F\}$) can implement the REP function, then the corollary follows using the Composition Lemma, the Hardness Lemma and the Lemmas 5 and 7. Otherwise, it follows from Lemma 6. ■

Lemma 14 *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. The* CSP($\mathcal{F}$) *is MAX SNP-hard.*

Proof: If either f_0 or f_1 is C-closed then using the Replication Lemma, we can implement the REP function and using the Composition Lemma along with Lemma 12 allows to conclude that CSP($\{f_0, f_1, f_2\}$) implements XOR function.

If neither f_0 nor f_1 is C-closed, then using the Unary Lemma, CSP($\{f_0, f_1, f_2\}$) can implement the unary functions $T(.)$ and $F(.)$, and then using the Composition Lemma along with Lemma 13, we conclude that CSP($\{f_0, f_1, f_2\}$) implements either the XOR function or the REP function. In the latter case, we can use Lemma 12 to conclude that CSP($\{f_0, f_1, f_2\}$) can implement the XOR function.

In either of the two situations above, we may conclude using Lemma 5 that CSP($\{f_0, f_1, f_2\}$) is MAX SNP-hard. ■

4 Hardness at Gap Location 1

It is possible to use a notion closely related to α-implementation to conclude from Schaefer's dichotomy theorem and 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 [11] who highlights the usefulness of such hardness results in other reductions.

An important characteristic of α implementation of a function f is that if we are given an assignment to the function variables which does not satisfy f, it can always be extended to the auxiliary variables such that precisely $(\alpha - 1)$ constraints are satisfied. This is a useful feature in establishing the hardness results for problems such as MAX 2-SAT which do not have hardness gaps located at 1. However, when dealing with problems with hardness gaps located at 1, such as MAX 3-SAT, it suffices to use a somewhat different notion of α-implementations, called *weak α-implementations*[†]. A weak α-implementation satisfies the condition (a)-(c) of the α-implementations and the condition (d) is replaced by the constraint that the CSP($\mathcal{F}$) instance implementing it has precisely α constraints. Clearly, weak α-implementations can be composed together and they preserve hardness gaps located at 1.

It is not difficult to verify that Schaefer's proof is in fact based on weak α-implementations of functions, and hence one may directly conclude from his proof that his class of NP-hard satisfiability problems are all in fact MAX SNP-hard. This yields Theorem 2.

5 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 to do so, it suffices to create a weak implementation of the function REP. Since given a weak 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 REP(X_1, X_2), REP(X_1, X_3), ..., REP(X_1, X_p). We now show how to create a weak implementation of the REP function; we need a definition :

Definition 15 [Weakly Positive and Weakly Negative Functions] *A function is called* weakly positive *(*weakly negative*) if it may be expressed as a* CNF *formula such that each clause has at most one negated (unnegated) variable.*

Now Lemmas 10 and 11 show that 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 weak implementation of the function REP or a weak implementation of both unary functions T and F. In the latter case, we can show the following lemma.

[†]The name weak α-implementation is slightly misleading because this notion is simultaneously both weaker and stricter than the notion of α-implementations.

Lemma 15 *If f is not weakly negative then* CSP$(\{f, T, F\})$ *can weak implement either the function $x \oplus y$, or the function $x + y$. Similarly, if f is not weakly positive then* CSP$(\{f, T, F\})$ *can weak 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 3 *If f_2 is not weakly positive and f_3 is not weakly negative, then* CSP$(\{f_2, f_3, T, F\})$ *weak 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 weak implement the XOR function. This, in turn, can be used to create a weak implementation of the function 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.

6 Conclusions

We initiated a study of the structure of a subclass of NPO and established an approximation dichotomy theorem for this class — in the process, giving a simple characterization to distinguish "easy" problems from the "hard" problems. Our dichotomy theorem strongly used the syntactic structure of problems in this class to identify a sharp division between the "easy" versus "hard" problems. We feel that it further stresses the role of syntactic prescriptions in the study of approximability. This was the fundamental insight on which the class MAX SNP was defined by Papadimitriou and Yannakakis [10]. Recent work of Khanna, Motwani, Sudan and Vazirani [7] also stresses this relationship, where they used syntactically defined classes to identify a structured core of hard problems for approximation classes such as APX, log-APX and poly-APX.

A natural question arising from this work is that of identifying larger classes which exhibit such dichotomies with respect to approximability. The syntactic optimization class MAX SNP contains our class of constraint satisfaction problems as a subclass. An interesting feature of this class is that every problem in this class is approximable to within a constant factor and there exist problems in this class which are MAX SNP-hard. Is it then the case that every problem in this class is either in P or MAX SNP-hard ? Feder and Vardi [5] recently initiated a systematic investigation to find the largest subclass of NP which exhibits a dichotomy for the decision version of the problems. Their efforts identify a subclass of SNP (the decision class underlying MAX SNP), called MMSNP - monotone, monadic SNP with no inequalities. This subclass contains constraint satisfaction problems over arbitrary domains and hence strictly contains Schaefer's class. They provide strong evidence that this class may in fact exhibit a P versus NP-hard dichotomy. We suspect that it is probably the case that the optimization analog of the class MMSNP exhibits a P versus MAX SNP-hard dichotomy.

In a recent related work, Khanna, Sudan and Williamson [8] have studied the optimization complexity of finding an assignment with maximum number of ones such that it satisfies every constraint in a given SAT$(\mathcal{F})$ problem. This class contains many natural optimization problems such as MAX CUT and MAX CLIQUE as its members. Surprisingly, this class exhibits an approximation hierarchy with essentially five levels. It contains problems which are in P or MAX SNP-hard or n^ϵ-hard and and yet does not have any intermediate approximation classes sandwiched in between.

Acknowledgments

We would like to thank Tomas Feder for providing us with a timely copy of the full version of [5] and to Christos Papadimitriou for pointing us to [4]. We would like to thank David Williamson for numerous discussions.

References

[1] S. ARORA AND S. SAFRA. Probabilistic checking of proofs: A new characterization of NP. *Proceedings of the* 33rd *Symposium on Foundations of Computer Science*, IEEE, 1992.

[2] S. ARORA, L. BABAI, J. STERN AND Z. SWEEDYK. The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. *Proceedings of the* 34th *Symposium on Foundations of Computer Science*, IEEE, 1993.

[3] 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.

[4] N. CREIGNOU. A Dichotomy Theorem for Maximum Generalized Satisfiability Problems. *To appear* in JCSS, December 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] U. FEIGE, S. GOLDWASSER, L. LOVASZ, S. SAFRA, AND M. SZEGEDY. Approximating clique is almost NP-complete. *Proceedings of the* 32nd *Symposium on Foundations of Computer Science*, IEEE, 1991.

[7] 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

[8] S. KHANNA, M. SUDAN, AND D.P. WILLIAMSON. The Optimization Complexity of Structure Optimization Problems. *In preparation*.

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

[10] C. PAPADIMITRIOU AND M. YANNAKAKIS. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43, pp. 425–440, 1991.

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

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

Appendix

A Schaefer's Theorem

We state Schaefer's dichotomy theorem in this section.

Definition 16 [Affine] *A function is said to be* affine *if it may be expressed as a system of linear equations of the form* $\sum_{i=1}^{k} x_i = 0$ *and* $\sum_{i=1}^{k} x_i = 1$ *(i.e. it evaluates to one iff the input variables satisfy the given equation system); the addition operation being modulo 2.*

Schaefer's dichotomy theorem may be stated as follows. Let $\mathcal{F}$ be a finite set of boolean functions. Then SAT($\mathcal{F}$) is always either in P or NP-hard. Furthermore, it is in P if and only if one of the following conditions is true:

1. Every $f \in \mathcal{F}$ is 0-valid.
2. Every $f \in \mathcal{F}$ is 1-valid.
3. Every $f \in \mathcal{F}$ is weakly positive.
4. Every $f \in \mathcal{F}$ is weakly negative.
5. Every $f \in \mathcal{F}$ is affine.
6. Every $f \in \mathcal{F}$ is bijunctive (i.e. expressible as a CNF formula with at most 2 literals per clause).

B Proof of Lemma 9 (The Hardness Lemma)

Proof: We prove this by using the Characterization Lemma for 2-monotone functions. 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 Characterization Lemma.

Suppose f violates the property (a) above. 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 $s_1 = 0^{p-b} 1^{q+b}$. Thus we have the following situation :

	$\overbrace{\quad}^{p-a}$	$\overbrace{\quad}^{a}$	$\overbrace{\quad}^{b}$	$\overbrace{\quad}^{q-b}$	$f()$
s	00...0	00...0	11...1	11...1	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	-

Observe that both a and b are non-zero. We consider the CSP($\{f, T, F, \text{REP}\}$) instance with the following set of constraints on variables $X_1, X_2, ..., X_k$:

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

It is now easy to verify that for $\alpha = (k-1)$, this instance α-implements the function $X_{p-a+1} \oplus X_{p+1}$ if $f(s_2) = 1$ and $X_{p-a+1}\overline{X_{p+1}}$, otherwise. The claim now follows immediately from Lemma 6.

Next suppose f violates the property (b) above. Then there exists an unsatisfying assignment s such that 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 be false. Consider one such unsatisfying assignment s. Without loss of generality, we have the following situation :

$$s \quad \overbrace{\underbrace{\overbrace{00\ldots0}^{V_1\setminus O(s)}}_{p}\ \underbrace{11\ldots1}_{q}\ \underbrace{\overbrace{11\ldots1}^{V_1\cap V_2}}_{r}}^{V_1}\ \overbrace{\underbrace{11\ldots1}_{s}\ \underbrace{\overbrace{00\ldots0}^{V_2\setminus O(s)}}_{t}}^{V_2}\ \underbrace{00\ldots0}_{u}\underbrace{11\ldots1}_{v}$$

We consider the CSP($\{f, T, F,$ REP$\}$) instance with the following set of constraints on variables $X_1, X_2, ..., X_k$:

- constraints REP(X_1, X_i) for $2 \leq i \leq p$,
- constraints $T(X_i)$ for $(p+1) \leq i \leq (p+q+r+s)$,
- constraints REP$(X_{p+q+r+s+1}, X_{p+q+r+s+i})$ for $2 \leq i \leq t$,
- constraints $F(X_i)$ for $(p+q+r+s+t+1) \leq i \leq (p+q+r+s+t+u)$,
- constraints $T(X_i)$ for $(p+q+r+s+t+u) \leq i \leq (p+q+r+s+t+u+v)$, and finally
- the constraint $f(X_1, X_2, ..., X_k)$ where $k = (p+q+r+s+t+u+v)$.

It is now easy to verify that for $\alpha = (k-1)$, this instance α-implements the function $X_1 + X_{p+q+r+s+1}$. Again, the claim now follows immediately from Lemma 6.

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