Constraint Satisfaction: The Approximability of Minimization Problems

Sanjeev Khanna[*] Madhu Sudan[†] Luca Trevisan[‡]

Abstract

This paper continues the work initiated by Creignou [5] and Khanna, Sudan and Williamson [15] who classify maximization problems derived from Boolean constraint satisfaction. Here we study the approximability of minimization *problems derived thence. A problem in this framework is characterized by a collection $\mathcal{F}$ of "constraints" (i.e., functions $f : \{0,1\}^k \to \{0,1\}$) and an instance of a problem is constraints drawn from $\mathcal{F}$ applied to specified subsets of n Boolean variables. We study the two minimization analogs of classes studied in [15]: in one variant, namely* MIN CSP $(\mathcal{F})$*, the objective is to find an assignment to minimize the number of unsatisfied constraints, while in the other, namely* MIN ONES $(\mathcal{F})$*, the goal is to find a satisfying assignment with minimum number of ones. These two classes together capture an entire spectrum of important minimization problems including s-t Min Cut, vertex cover, hitting set with bounded size sets, integer programs with two variables per inequality, graph bipartization, clause deletion in CNF formulae, and nearest codeword. Our main result is that there exists a finite partition of the space of all constraint sets such that for any given $\mathcal{F}$, the approximability of* MIN CSP $(\mathcal{F})$ *and* MIN ONES $(\mathcal{F})$ *is completely determined by the partition containing it. Moreover, we present a compact set of rules that determines which partition contains a given family $\mathcal{F}$. Our classification identifies the central elements governing the approximability of problems in these classes, by unifying a large collection algorithmic and hardness of approximation results. When contrasted with the work of [15], our results also serve to formally highlight inherent differences between maximization and minimization problems.*

[*] Fundamental Mathematics Research Department, Bell Labs, 700 Mountain Avenue, NJ 07974. sanjeev@research.bell-labs.com.

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

[‡] Centre Universitaire d'Informatique, Université de Genève, Rue Général-Dufour 24, CH-1211, Genève, Switzerland. Work done at the University of Rome "La Sapienza". trevisan@cui.unige.ch.

1. Introduction

In this paper we present a complete classification of the approximability of minimization problems derived from "Boolean constraint satisfaction". Our work follows the work of Creignou [5] and Khanna, Sudan and Williamson [15] who obtained such a classification for maximization problems.

This line of research is motivated by an attempt to unify the many known positive and negative results on the approximability of combinatorial optimization problems. In the case of positive results, many paradigms have been obtained and these serve to unify the results nicely. In contrast, there is a lack of similar unification among negative results. Part of the reason for this is that hardness results tipically tend to exploit every feature of the problem whose hardness is being shown, rather than isolating the "minimal" features that would suffice to obtain the hardnes result. As a result many interesting questions about hard problems tend to remain unresolved. Khanna et al. [15] describe a number of such interesting questions: (1) Are there any NP-hard problems in MAX SNP which are not MAX SNP-hard? (2) Are there any "natural" maximization problems which are approximable to within polylogarithmic factors, but no better? (3) Is there some inherent difference between maximization and minimization problems among combinatorial optimization problems?

In order to study such questions, or even to place them under a formal setting, one needs to first specify the optimization problems one wishes to study in some uniform framework. Furthermore, one has to be careful to ensure that it is possible to "decide" whether the optimization problem studied is easy or hard (to, say, compute exactly). Unfortunately, barriers such as Rice's theorem (which says this question may not in general be decidable) or Ladner's theorem (which says problems may not be just easy or hard [20]) force us to severely restrict the class of problems which can be studied.

A work of Schaefer [25] from 1978 isolates one class of decision problems which can actually be classified completely. He obtains this classification by restricting his attention to "Boolean constraint problems". A typical problem in this class is defined by a finite set $\mathcal{F}$ of finite Boolean constraints (specified by, say, a truth table). An instance of such a problem specifies m "constraint applications" on n Boolean variables where each constraint application is the application of one of the constraints from $\mathcal{F}$ to some subset (actually, ordered tuple would be more exact) of the n variables. The language SAT($\mathcal{F}$) consists of all instances which have an assignment satisfying all m constraints. Schaefer describes six classes of function families, such that if $\mathcal{F}$ is a subset of one of these classes, then the decision problem is in P, else he shows that the decision problem is NP-hard.

Creignou [5] and Khanna et al. [15] extend the study above, in a natural way, to optimization problems. They define two classes of optimization problems: MAX CSP ($\mathcal{F}$) and MAX ONES ($\mathcal{F}$) (Actually the work of Creignou's studies only the class MAX CSP ($\mathcal{F}$).). The instances in both cases are m constraints applied on n Boolean variables, where the constraints come from $\mathcal{F}$. In the former case, the objective is to find an assignment which maximizes the number of constraints that are satisfied. In the latter case, the objective is to find an assignment to the Boolean variables which satisfies all the constraints while maximizing the weight of the assignment (i.e., the number of variables set to 1). In a result similar to that of Schaefer's they show that there exists a finite partition of the space of all function families such that the approximability of a given problem is completely determined based on which partition the family $\mathcal{F}$ belongs to. The interesting aspect of this classification result is that it manages to capture diverse problems such as MAX FLOW, MAX CUT and MAX CLIQUE (which are all approximable to very different factors) and yet unifies the (non)-approximability results for all such problems. Within the framework of constraint satisfaction problems, Khanna et al. settle the questions (1) and (2) raised above. Our work is directed towards question (3).

We consider the two corresponding classes of minimization problems which we call MIN CSP ($\mathcal{F}$) and MIN ONES ($\mathcal{F}$). Again, instances of both problems consist of m constraints from $\mathcal{F}$ applied to n Boolean variables. The objective in MIN CSP ($\mathcal{F}$) is to find the assignment which minimizes the number of unsatisfied constraints. The objective for MIN ONES ($\mathcal{F}$) is to find the assignment which satisfies all constraints while minimizing the number of the variables set to 1. For each class of optimization problems our main theorem is informally stated as follows: There exists a finite partition of the space of all function families, such that the approximability of the problem MIN CSP ($\mathcal{F}$) (resp. MIN ONES ($\mathcal{F}$)) is determined completely by which partition it lies in. We stress however that there is one important respect in which our classification is different from previous ones. Our partitions include several classes whose approximability is still not completely understood. Thus while our result shows that the number of "distinct" levels of approximability (among minimization problems derived from constraint satisfaction) is finite — it only places an upper bound on the number of levels — it is unable to pin it down exactly. By pinning down a complete problem for each partition, we, however turn this seeming weakness into a strength by highlighting some important problems whose approximability deserves further attention.

Even though the transition from maximization problems to minimization problems is an obvious next step, success in this transition is not immediate. For starters — the transition from SAT to MAX CSP is completely analogous to the transition from SNP to MAX SNP. Yet, there is no minimization analog of MAX SNP. The obvious difficulty seems to be that we are immediately confronted by a host of problems for which distinguishing the case where the optimum is zero, from the case for which the optimum is non-zero is NP-hard. The traditional approach to deal with zero/one problem has been to restrict the syntax using which the predicate within the SNP construct is used - thereby ruling out the hardness of the zero/one problem (see e.g. [18, 19]). Our approach, via constraint satisfaction, however does not place any such restrictions. We simply characterize all the problems for which the 0/1 problem is hard, and then having done so, move to the rest of the problems. All the different levels of approximability that are seen emerge naturally.

Despite this completely oblivious approach to defining the classes MIN CSP and MIN ONES the classes end up capturing numerous natural optimization problems — with very distinct levels of approximability. For starters, the s-t MIN CUT problem is one of the problems captured by MIN CSP which is well known to be computable exactly in P. (This was already shown and used by Khanna et al. [15].) At the constant level of approximability we see problems such as VERTEX COVER [11, 22], Hitting Set with bounded size sets [13], Integer programs with two variables per inequality [12]. (The references cited after the problems show that the problem is approximable to within constant factors.) Then we come to two open problems: MIN UNCUT [10] and MIN 2CNF DELETION [17] both of which are known to be approximable to within polylogarithmic factors and known to be hard to approximate to within some constant factor. The exact approximability of both problems remains open. At a higher level of approximability is the NEAREST CODEWORD problem [1] which is known to be approximable to within polynomial factors but is hard to approximate to within $2^{\log^{\epsilon} n}$ factors. For each of these problems we show that there is a constraint family $\mathcal{F}$ such that either MIN CSP $(\mathcal{F})$ or MIN ONES $(\mathcal{F})$ is isomorphic to the problem. The ability to study all these different problems in a uniform framework and extract the features that make the problems easier/harder than the others shows the advantage of studying optimization problems under the constraint satisfaction framework.

Lastly, we point out that it is not only the negative results that are unified by our framework but also the positive results. Our positive results highlight once more the utility of the linear programming (LP) relaxation followed by rounding approach to devising approximation algorithms. This approach, which plays a significant role in all the above mentioned results of [22, 13, 12, 10, 17], also plays a crucial role in obtaining constant factor approximation algorithms for one of the partitions of the MIN CSP $(\mathcal{F})$ problems and one partition of the MIN ONES $(\mathcal{F})$ problems.

One limitation of our results is that they focus on problems in which the input instances have no restrictions in the manner in which constraints may be imposed on the input variables. This is the reason why many of the problems turn out to be as hard as shown. Sometimes significant insight may be gleaned from restricting the problem instances. A widely prescribed condition is that the incidence graph on the variables and the constraints should form a planar graph. This restriction has been recently studied by Khanna and Motwani [14] and they show that it leads to polynomial time approximation schemes for a general class of constraint satisfaction problems. Another input restriction of interest could be that variables are allowed to participate only in a bounded number of constraints. We are unaware of any work on this front. An important extension of our work would be to consider constraint families which contain constraints of unbounded arity (such as those considered in $\mathsf{MIN}F^{+}\Pi_1$). Such an extension would allow us to capture problems such as SET COVER. In summary, our work reflects yet another small step towards the big goal of understanding the structure of optimization problems.

2. Preliminaries

The notion of constraints and constraint applications and our classes of problems of interest have already been defined informally above. We formalize them in the next two subsections. We next review some basic concepts and definitions in approximability, reductions and completeness. Finally, we present our classification theorems and give an overview of how the remainder of this paper is organized.

2.1 Constraints, Constraint Applications and Constraint Families

A constraint is a function $f : \{0,1\}^k \rightarrow \{0,1\}$. A constraint application is a pair $\langle f, (i_1, \ldots, i_k)\rangle$, 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 $j \neq j'$. A contraint family $\mathcal{F}$ is a finite collection of constraints $\{f_1, \ldots, f_l\}$. Constraints and constraint families are the ingredients that specify an optimization problem. Thus it is necessary that their description be finite. Constraint applications are used to specify instances of optimization problems and the fact that their description lengths grow with the instance size is crucially exploited here. While this 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 $C = \langle f, (i_1, \ldots, i_k)\rangle$ 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". We now describe the optimization problems considered in this paper.

Definition 1 (MIN CSP $(\mathcal{F})$**)**
INPUT : *A collection of m constraint applications of the form $\{\langle f_j, (i_1(j), \ldots, i_{k_j}(j))\rangle\}_{j=1}^{m}$, 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 minimize the number of unsatisfied constraints.*

In the weighted problem MIN WEIGHT CSP $(\mathcal{F})$ *the input includes m non-negative weights $w_1 . \ldots, w_m$ and the objective is to find an assignment which minimizes the sum of the weights of the unsatisfied constraints.*

Definition 2 (MIN ONES $(\mathcal{F})$**)**
INPUT : *A collection of m constraint applications of the form $\{\langle f_j, (i_1(j), \ldots, i_{k_j}(j))\rangle\}_{j=1}^{m}$, 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 which satisfies all the constraints and minimizes the total number of variables assigned true.*

In the weighted problem MIN WEIGHT ONES $(\mathcal{F})$ *the input includes n non-negative weights $w_1 . \ldots, w_n$ and the objective is to find an assignment which satisfies all constraints and minimizes the sum of the weights of variables assigned to* 1.

Representation of functions We will often work with the *maxterm* representation of functions:

Definition 3 [Maxterm] *Given a function $f(x_1, x_2, ..., x_k)$, a subset of literals defined over the variables x_i's is called a* maxterm *if setting each of* *the literals false determines the function to be false and if it is a minimal such collection.*

We express a maxterm $m = \{l_1, l_2, ..., l_m\}$ where each l_j is x_i or $\bar{x}_i$ for some x_i, as $\vee_{j=1}^{m} l_j$. Thus if $m_1, m_2, ..., m_q$ are all the maxterms of a function f, then f may be represented as $\bigwedge_{i=1}^{q} m_i$. This is called a maxterm representation of a function f.

Properties of function families We now describe the main properties that are used to classify the approximability of the optimization problems. The approximability of a function family is determined by which of the properties the family satisfies. We start with the six properties defined by Schaefer:

- A constraint f is *0-valid* (resp. *1-valid*) if $f(0, \ldots, 0) = 1$ (resp. $f(1, \ldots, 1) = 1$).
- A constraint is *weakly positive* (resp. *weakly negative*) if it can be expressed as a CNF-formula having at most one negated variable (resp. at most one unnegated variable[1]) in each clause.
- A constraint is *affine* if it can be expressed as a conjunction of linear equalities over $\mathcal{Z}_2$.
- A constraint is *2cnf* if it is expressible as a 2CNF-formula (that is, a CNF formula with at most two literals per clause).

The above definitions extend to constraint families naturally. For instance, a constraint family $\mathcal{F}$ is *0-valid* if *every* constraint $f \in \mathcal{F}$ is 0-valid. Using the above definitions Schaefer's theorem may be stated as follows: For any constraint family $\mathcal{F}$, SAT$(\mathcal{F})$ is in P if $\mathcal{F}$ is 0-valid or 1-valid or weakly positive or weakly negative or affine or 2cnf; else deciding SAT$(\mathcal{F})$ is NP-hard.

Some more properties were defined by Khanna et al. [15] to describe the approximability of the problems they considered. We will need them for our results as well.

- f if *2-monotone* if $f(x_1, \ldots, x_k)$ is expressible as $(x_{i_1} \bigwedge \cdots \bigwedge x_{i_p}) \bigvee (\neg x_{j_1} \bigwedge \cdots \bigwedge \neg x_{j_q})$ (i.e., f is expressible as a DNF-formula with at most two terms - one containing only positive literals and the other containing only negative literals).
- A constraint is *width-2 affine* if it is expressible as a conjunction of linear equations over $\mathcal{Z}_2$ such that each equation has at most 2 variables.
- A constraint f is *C-closed* if for all assignments s, $f(s) = f(\bar{s})$.

The above properties, along with Schaefer's original set of properties suffice for [5] and [15] to classify the approximability of the maximization problems MAX CSP $(\mathcal{F})$ and MAX ONES $(\mathcal{F})$. A statement of their results is included in Appendix A.

[1] Such clauses are usually called Horn clauses.

Lastly we need one definition of our own, before we can state our results.

- A constraint f is *IHS-B^+* (for *Implicative Hitting Set-Bounded+*) if it is expressible as a CNF formula where the clauses are of one of the following types: $x_1 \bigvee \cdots \bigvee x_k$ for some positive integer k, or $\neg x_1 \bigvee x_2$, or $\neg x_1$. IHS-B^- constraints and constraint families are defined analogously (with every literal being replaced by its complement). A family is a IHS-B family if the family is a IHS-B^+ family or a IHS-B^- family.

Problems captured by MIN CSP **and** MIN ONES We enumerate here some interesting minimization problems which are "captured" by (i.e., are equivalent to some problem in) MIN CSP and MIN ONES. The following list is interesting for several reasons. First, it highlights the importance of the classes MIN CSP and MIN ONES as classes that contain interesting minimization problems. Furthermore, these problems turn out to be "complete" problems for the partitions they belong to - thus they are necessary for a full statement of our results. Last, for several of the problems listed below, their approximability is far from being well-understood. We feel that these problems are somehow representative of the lack of our understanding of the approximability of minimization problems.

- The well-known Hitting Set problem, when restricted to sets of bounded sizes B can be captured as MIN ONES$(\mathcal{F})$ for $\mathcal{F} = \{x_1 \bigvee \cdots \bigvee x_k | k \leq B\}$. Also, of interest to our paper is a slight generalization of this problem which we call the Implicative Hitting Set-B Problem (MIN IHS-B) which is MIN CSP$(\mathcal{F})$ for $\mathcal{F} = \{x_1 \bigvee \cdots \bigvee x_k : k \leq B\} \cup \{\neg x_1 \bigvee x_2\} \cup \{\neg x_1\}$. The MIN ONES version of this problem will be of interest to us as well. The Hitting Set-B problem is well-known to be approximable to within a factor of B. We show that, in fact MIN IHS-B is approximable to within a factor of $B + 1$.
- MIN UNCUT = MIN CSP$(\{x \oplus y = 1\})$. This problem has been studied previously by Klein et al. [16] and Garg et al. [10]. The problem is known to be MAX SNP-hard and hence not approximable to within a constant factor. On the other hand, the problem is known to be approximable to within a factor of $O(\log n)$ [10].
- MIN 2CNF DELETION = MIN CSP$(\{x \bigvee y, \neg x \bigvee \neg y\})$. This problem has been studied by Klein et al. [17]. They show that

the problem is MAX SNP-hard and that it is approximable to within a factor of $O(\log n \log\log n)$.

- NEAREST CODEWORD = MIN CSP($\{x \oplus y \oplus z = 0, x \oplus y \oplus z = 1\}$). This is a classical problem for which hardness of approximation results have been shown by Arora et al. [1]. The MIN ONES version of this problem is essentially identical to this problem. For both problems, the hardness result of Arora et al. [1] says that approximating this problem to within a factor of $2^{\log^{\epsilon} n}$ is hard, unless NP $\subset$ QP. No non-trivial approximation guarantees are known for this problem (the trivial bound being a factor of m, which is easily achieved since deciding if all equations are satisfiable amounts to solving a linear system).
- Lastly we also mention one more problem which is required to present our main theorem. MIN HORN DELETION $=$ MIN CSP($\{x, \neg x, (\neg x \bigvee y \bigvee z)\}$). This problem is essentially as hard as the NEAREST CODEWORD.

2.2 Approximability, Reductions and Completeness

Finally, before presenting our results, we mention some basic notions on approximability. A *combinatorial optimization* problem is defined over a set of *instances* (admissible input data); a finite set $\mathsf{sol}(x)$ of *feasible solutions* is associated to any instance. An *objective function* attributes an integer value to any solution. The *goal* of an optimization problem is, given an instance x, find a solution $y \in \mathsf{sol}(x)$ of *optimum* value. The optimum value is the largest one for *maximization* problems and the smallest one for *minimization* problems. A combinatorial optimization problem is said to be an NPO problem if instances and solutions are easy to recognize, solutions are short, and the objective function is easy to compute. See e.g. [4] for formal definitions.

Definition 4 (Performance Ratio) *An approximation algorithm for an* NPO *problem A has* performance ratio $\mathcal{R}(n)$ *if, given any instance $\mathcal{I}$ of A with $|\mathcal{I}| = n$, it computes a solution of value V which satisfies*

$$\max\left\{\frac{V}{\mathsf{opt}(\mathcal{I})}, \frac{\mathsf{opt}(\mathcal{I})}{V}\right\} \leq \mathcal{R}(n).$$

A solution satisfying the above inequality is referred to as being $\mathcal{R}(n)$*-approximate*. We say that a NPO problem is approximable to within a factor $\mathcal{R}(n)$ if it has a polynomial-time approximation algorithm with performance ratio $\mathcal{R}(n)$.

Definition 5 (Approximation Classes) *An* NPO *problem A is in the class* PO *if it is solvable to optimality in polynomial time. A is in the class* APX *(resp.* log-APX/ poly-APX*) if there exists a polynomial-time algorithm for A whose performance ratio is bounded by a constant (resp. logarithmic/polynomial factor in the size of the input).*

Completeness in approximation classes can be defined using appropriate approximation preserving reducibilities. These reducibilities tend to be a bit subtle and we will be careful to specify the reducibilities used in this paper. In this paper, we heavily use two notions of reducibilites defined below. (1) A-reducibility which ensures that if Π is A-reducible to Π' and Π' is $r(n)$ approximable for some function $r : \mathcal{Z}^+ \rightarrow \mathcal{Z}^+$, then Π is $\alpha r(n^c)$-approximable, for some constants α and c. In particular if Π' is approximable to within some constant factor (resp. $O(\log n)$, $n^{O(1)}$ factor), then Π is also approximable to within some constant factor (resp. $O(\log n)$, $n^{O(1)}$ factor). (2) AP-reducibility which is a more stringent notion of reducibility, in that every AP-reduction is also an A-reduction This reducibility has the feature that if Π AP-reduces to Π' and Π' has a PTAS, then Π has a PTAS. Unfortunately neither one of these reducibilities alone suffices for our purposes — we need to use the more stringent reducibility to show APX-hardness of problems and we need the flexibility of the weaker reducibility to provide the other hardness results. Fortunately, results showing APX-hardness follow directly from [15] and so the new reductions of this paper are all A-reductions.

Definition 6 (AP-reducibility [6]) *For a constant $\alpha > 0$ and two* NPO *problems A and B, we say that A is AP-reducible to B if two polynomial-time computable functions f and g exist such that the following holds:*

(1) *For any instance $\mathcal{I}$ of A, $f(\mathcal{I})$ is an instance of B.*

(2) *For any instance $\mathcal{I}$ of A, and any feasible solution $\mathcal{S}'$ for $f(\mathcal{I})$, $g(\mathcal{I}, \mathcal{S}')$ is a feasible solution for x.*

(3) *For any instance $\mathcal{I}$ of A and any $r > 1$, if $\mathcal{S}'$ is an r-approximate solution for $f(\mathcal{I})$, then $g(\mathcal{I}, \mathcal{S}')$ is an $(1 + (r-1)\alpha + o(1))$-approximate solution for $\mathcal{I}$, where the o notation is with respect to $|\mathcal{I}|$.*

We say that A is AP-reducible to B if a constant $\alpha > 0$ exists such that A is α-AP-reducible to B.

Definition 7 (A-reducibility [7]) *An* NPO *problem A is said to be A-reducible to an* NPO *problem B if two polynomial time computable functions f and g and a constant α exist such that:*

(1) *For any instance $\mathcal{I}$ of A, $f(\mathcal{I})$ is an instance of B.*

(2) *For any instance $\mathcal{I}$ of A and any feasible solution $\mathcal{S}'$ for $f(\mathcal{I})$, $g(\mathcal{I}, \mathcal{S}')$ is a feasible solution for $\mathcal{I}$.*

(3) *For any instance $\mathcal{I}$ of A and any $r > 1$, if $\mathcal{S}'$ is a r-approximate solution for $f(\mathcal{I})$ then $g(\mathcal{I}, \mathcal{S}')$ is an (αr)-approximate solution for $\mathcal{I}$.*

Remark 8 *The original definitions of AP-reducibility and A-reducibility were more general. Under the original definitions, the A-reducibility does not preserve membership in* log-APX*, and it is not clear whether every AP-reduction is also an A-reduction. The restricted versions defined here are more suitable for our purposes. In particular, it is true that the Vertex Cover problem is* APX*-complete under our definition of AP-reducibility.*

Definition 9 (APX **and** poly-APX**-completeness)** *An* APX *problem A is* APX*-complete if any* APX *problem is AP-reducible to A. A* poly-APX *problem A is* poly-APX*-complete if any* poly-APX *problem is A-reducible to A.*

It is easy to prove that if A is APX-complete (resp. poly-APX-complete) then a constant ϵ exists such that it is NP-hard to approximate A within $(1+\epsilon)$ (resp. n^{ϵ}).

One of our hardness result will be proved by means of a reduction from the MIN TOTAL LABEL-COVER problem, defined as follows.

Definition 10 (MIN TOTAL LABEL-COVER**)** *An instance of the* MIN TOTAL LABEL-COVER *problem contains integer parameters Q_1, Q_2, A_1, A_2, and R; and functions*

$$q_1 : [R] \to Q_1 \ , \ q_2 : [R] \to Q_2 \ ,$$
$$V : [R] \times [A_1] \times [A_2] \to \{0,1\}$$

A feasible solution is a pair of functions p_1, p_2, where $p_1 : [Q_1] \to 2^{[A_1]}$ and $p_2 : [Q_2] \to 2^{[A_2]}$, such that for every $r \in [R]$, there exists $a_1 \in p_1(q_1(r))$ and $a_2 \in p_2(q_2(r))$ such that $V(r, a_1, a_2) = 1$. The objective function to be minimized is $\sum_{q_1 \in Q_1} |p_1(q_1)| + \sum_{q_2 \in Q_2} |p_2(q_2)|$.

This is a variation, introduced by Amaldi and Kann, of the MIN LABEL-COVER problem [21, 1] (in the MIN LABEL-COVER problem the objective function to be minimized is $\sum_{q_1 \in Q_1} |p_1(q_1)|$). A reduction from the multi-prover proof-systems of [24, 2] shows that, for any $\epsilon > 0$, it is NP-hard to approximate MIN TOTAL LABEL-COVER within $2^{\log^{1-\epsilon} n}$. The reduction in question is similar to the standard one from multi-prover proof systems to MIN LABEL-COVER [21, 1] and omitted from this extended abstract.

2.3 Main Results

We now present the main results of this paper. The theorem uses the shorthand Π' is Π-complete to indicate that the problem Π' is equivalent (under A-reductions) to the problem Π.

Theorem 11 (MIN CSP **Classification)** *For every constraint set $\mathcal{F}$,* MIN CSP$(\mathcal{F})$ *is either in* PO *or* APX*-complete or* MIN UNCUT*-complete or* MIN 2CNF DELETION*-complete or* NEAREST CODEWORD*-complete or* MIN HORN DELETION*-complete or the decision problem is* NP*-hard. Furthermore,*

(1) *If $\mathcal{F}$ is 0-valid or 1-valid or 2-monotone, then* MIN CSP$(\mathcal{F})$ *is in* PO.

(2) *Else if $\mathcal{F}$ is IHS-B then* MIN CSP$(\mathcal{F})$ *is* APX*-complete.*

(3) *Else if $\mathcal{F}$ is width-2 affine then* MIN CSP$(\mathcal{F})$ *is* MIN UNCUT*-complete.*

(4) *Else if $\mathcal{F}$ is 2CNF then* MIN CSP$(\mathcal{F})$ *is* MIN 2CNF DELETION*-complete.*

(5) *Else if $\mathcal{F}$ is affine then* MIN CSP$(\mathcal{F})$ *is* NEAREST CODEWORD*-complete.*

(6) *Else if $\mathcal{F}$ is weakly positive or weakly negative then* MIN CSP$(\mathcal{F})$ *is* MIN HORN DELETION*-complete.*

(7) *Else deciding if the optimum value of an instance of* MIN CSP$(\mathcal{F})$ *is zero is* NP*-complete.*

Theorem 12 (MIN ONES **Classification)** *For every constraint set $\mathcal{F}$,* MIN ONES $(\mathcal{F})$ *is either in* PO *or* APX*-complete or* NEAREST CODEWORD*-complete or* MIN HORN DELETION*-complete or* poly-APX*-complete or the decision problem is* NP*-hard. Furthermore,*

(1) *If $\mathcal{F}$ is 0-valid or weakly negative or affine with width 2, then* MIN ONES $(\mathcal{F})$ *is in* PO.

(2) *Else if $\mathcal{F}$ is 2CNF or IHS-B then* MIN ONES $(\mathcal{F})$ *is* APX*-complete.*

(3) *Else if $\mathcal{F}$ is affine then* MIN ONES $(\mathcal{F})$ *is* NEAREST CODEWORD*-complete.*

(4) *Else if $\mathcal{F}$ is weakly positive then* MIN ONES $(\mathcal{F})$ *is* MIN HORN DELETION*-complete.*

(5) *Else if $\mathcal{F}$ is 1-valid then* MIN ONES $(\mathcal{F})$ *is* poly-APX *complete*

(6) *Else finding any feasible solution to* MIN ONES $(\mathcal{F})$ *is* NP*-hard.*

Techniques As in the work of Khanna et al. [15] two simple ideas play an important role in this paper. (1) The notion of *implementations* from [15] (also known as *gadgets* [3, 26]) which shows how to use the constraints of a family $\mathcal{F}$ to enforce constraints of a different family $\mathcal{F}'$, thereby laying the groundwork of a reduction from MIN CSP$(\mathcal{F}')$ to MIN CSP$(\mathcal{F})$. (2) The idea of working with weighted versions of minimization problems. Even though our theorems only make statements about unweighted versions of problems, all our results use as intermediate steps the weighted versions of these problems. The weights allow us to manipulate problems more locally. However, simple and well-known ideas eventually allow us to get rid of the weights and thereby yielding hardness of the unweighted problem as well. As a side-effect we also show (in Section 3.2) that the unweighted and weighted problems are equally hard to approximate in all the relevant cases of MIN CSP and MIN ONES problems. This extends to minimization problems the results of Crescenzi et al. [8].

A more detailed look at implementations and weighted problems follows in Section 3. In Section 4 we show the containment results for the MIN CSP result. The new element here is the constant factor approximation algorithm for IHS-B families. In Section 5 we show the hardness results. The new element here is the characterization of functions which are not expressible as IHS-B and the MIN HORN DELETION-completeness results for weakly positive and negative families. We show a close correspondence between MIN CSP and MIN ONES problems in Section 6. Finally, in Sections 7 and 8, we give our positive and negative results for MIN ONES problems.

3. Warm-up

3.1 Implementations

Suppose we want to show that for some constraint set $\mathcal{F}$, the problem MIN ONES$(\mathcal{F})$ is APX-hard. We will start with a problem that is known to be APX-hard, such as VERTEX COVER, which is the same as MIN ONES$(\{x \bigvee y\})$. We will then have to reduce this problem to MIN ONES$(\mathcal{F})$. The main technique we use to do this is to "implement" the constraint $x \bigvee y$ using constraints from the constraint set $\mathcal{F}$. The following definition shows how to formalize this notion. (The definition is part of a more general definition of Khanna et al [15]. In fact, their definition is needed for AP-reductions, but since we don't provide any new AP-reductions, we don't need their full definition here.)

Definition 13 (Perfect Implementation [15]) *A collection of constraint applications $C_1, \ldots, C_\alpha$ over a set of variables $\vec{x} = \{x_1, x_2, ..., x_p\}$ and $\vec{y} = \{y_1, y_2, ..., y_q\}$ is called a* perfect α-implementation *of a constraint $f(\vec{x})$ iff the following conditions are satisfied:*

(1) *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 all the constraints are satisfied,*

(2) *For any assignment of values to $\vec{x}$ such that $f(\vec{x})$ is false, no assignment of values to $\vec{y}$ can satisfy all the constraints.*

A constraint set $\mathcal{F}$ perfectly implements a constraint f if there exists a 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 perfectly. It is easily seen that perfect implementations compose together, i.e., if $\mathcal{F}_f$ perfectly implements f, and $\mathcal{F}_g$ perfectly implements $g \in \mathcal{F}_f$, then $(\mathcal{F}_f \setminus \{g\}) \cup \mathcal{F}_g$ perfectly implements f. In order to see the utility of implementations, it is better to work with weighted problems.

3.2 Weighted Problems

For a function family $\mathcal{F}$, the problem MIN WEIGHT CSP$(\mathcal{F})$ has as instances m weighted constraints $C_1, \ldots, C_m$ with non-negative weights $w_1, \ldots, w_m$ on n Boolean variables $x_1, \ldots, x_n$. The objective is to find an assignment to $\vec{x}$ which minimizes the weight of unsatisfied constraints. An instance of the problem MIN WEIGHT ONES$(\mathcal{F})$ has as instances m constraints $C_1, \ldots, C_m$ on n weighted Boolean variables $x_1, \ldots, x_n$ with non-negative weights $w_1, \ldots, w_n$. The objective is to find the assignment which minimizes the sum of weights of variables set to 1 among all assignments that satisfy all constraints. The following proposition shows how implementations are useful for reductions among weighted problems.

Proposition 14 *If a constraint family $\mathcal{F}'$ perfectly implements every function $f \in \mathcal{F}$, then* MIN CSP$(\mathcal{F})$ *(resp.* MIN WEIGHT CSP, MIN WEIGHT ONES$(\mathcal{F})$*) is A-reducible to* MIN CSP$(\mathcal{F}')$ *(resp.* MIN WEIGHT CSP, MIN WEIGHT ONES$(\mathcal{F}')$*).*

Proof: Let k be large enough so that any constraint from $\mathcal{F}$ has a perfect k-implementation using constraints from $\mathcal{F}'$. Let $\mathcal{I}$ be an instance of MIN WEIGHT CSP$(\mathcal{F})$ and let $\mathcal{I}'$ be the instance of MIN WEIGHT CSP$(\mathcal{F}')$ obtained by replacing each constraint of $\mathcal{I}$ with the respective k-implementation. It is easy to check that any assignment for $\mathcal{I}'$ of cost V yields an assignment for $\mathcal{I}$ whose cost is between V/k and V. It is immediate to check that if the former solution is r-approximate, then the latter is (kr)-approximate. □

While weighted problems allow for the convenient use of implementations, there is really not much of a difference between weighted and unweighted problems. It is easy to show that MIN WEIGHT ONES$(\mathcal{F})$ A-reduces to MIN ONES$(\mathcal{F})$. It is also easy to see that if we are allowed to repeat the same constraint many times, then MIN WEIGHT CSP$(\mathcal{F})$ A-reduces to MIN CSP$(\mathcal{F})$. Finally, it turns out that the equivalence holds even when we are not allowed to repeat constraints. This is summarized in the following Theorem.

Theorem 15 (Weight-removing Theorem) *For any constraint family $\mathcal{F}$,* MIN WEIGHT ONES$(\mathcal{F})$ *A-reduces to* MIN ONES$(\mathcal{F})$. *If $\mathcal{F}$ perfectly implements $(x = y)$, then* MIN WEIGHT CSP$(\mathcal{F})$ *A-reduces to* MIN CSP$(\mathcal{F})$.

As a first step towards establishing this result, we recall that from the results of [8], it follows that whenever MIN WEIGHT CSP$(\mathcal{F})$ (resp. MIN WEIGHT ONES$(\mathcal{F})$) is in poly-APX, then it is AP-reducible (and hence A-reducible) to the restriction where weights are polynomially bounded (in particular, they can be assumed to be bounded by $\max\{n^2, m^2\}$, where m is the number of constraints and n the number of variables). For this reason, from now on, weighted problems will always be assumed to have polynomially bounded weights. Moreover, in a MIN WEIGHT CSP$(\mathcal{F})$ instance, we will sometimes see a weighted constraint of weight w as a collection of w identical constraints.

In a MIN WEIGHT CSP instance we can assume that no constraint has weight zero (otherwise we can remove the constraint without changing the problem). We also assume that in a MIN WEIGHT ONES instance no variable has weight zero. Otherwise, we multiply all the weights by n^2 ($n =$ number of variables) and then we change the zero-weights to 1. This negligibly perturbs the problem and gives an AP-reduction. This is formalized below.

Proof of Theorem 15: We begin by showing that for any family $\mathcal{F}$, MIN WEIGHT CSP$(\mathcal{F})$ AP-reduces to MIN CSP$(\mathcal{F} \cup \{(x = y)\})$. For this, we use an argument similar to the reduction from MAX 3SAT to MAX 3SATB (see [23]), however we don't need to use expanders. Let $\mathcal{I}$ be an instance of MIN WEIGHT CSP$(\mathcal{F})$ over variable set $X = \{x_1, \ldots, x_n\}$. For any $i \in [n]$, let occ_i be the number of the constraints where x_i appears. We make occ_i "copies" of x_i, and call them $y_i^1, \ldots, y_i^{occ_i}$. We substitute the j-th occurrence of x_i by y_i^j. We repeat this substitution for any variable. Additionally, for $i \in [n]$, we add all the possible $occ_i(occ_i - 1)/2$ "consistency" constraints of the form $y_i^j = y_i^h$ for $j, h \in [occ_i]$, $i \neq j$. Call $\mathcal{I}'$ the new instance; observe that $\mathcal{I}'$ contains no repetition of constraints. Moreover, any assignment $\vec{a}$ for $\mathcal{I}'$ can be converted into an assignment $\vec{a}'$ that satisfies all the consistency constraints without increasing the cost. Indeed, if, for some i, not all the y_i^h have the same value under $\vec{a}$, then we give value 0 to all of them. This can, at most, contradict all the constraints containing an occurrence of a switched variable, but this satisfies many more consistency constraints than those that got contradicted.

We next show that for any family $\mathcal{F}$, MIN WEIGHT ONES$(\mathcal{F})$ AP-reduces to MIN ONES$(\mathcal{F})$. To begin with, note that if MIN WEIGHT ONES$(\mathcal{F})$ is in PO, then it is trivially AP-reducible to any NPO problem (including, in particular, MIN ONES$(\mathcal{F})$). The interesting case thus arises when $\mathcal{F}$ is not 0-valid nor width-2 affine nor weakly negative. As can be seen from the proof of Lemma 48 below, in such case either $\mathcal{F}$ perfectly implements $(x = y)$ or all the basic constraints of $\mathcal{F}$ are of the form $x_1 \bigvee \cdots \bigvee x_k$ for some $k \geq 1$.

If $\mathcal{F}$ perfectly implements $x = y$, then for any variable x_i of weight w_i we introduce $w_i - 1$ new variables $y_i^1, \ldots, y_i^{w_i - 1}$ and the implementations of the constraints $x_i = y_i^1$, $y_i^1 = y_i^2$, $\ldots$, $y_i^{w_1 - 1} = x_i$. Each variable has now cost 1. Any solution satisfying the original set of constraints can be converted into a solution for the new set of constraints by letting $y_i^j = x_i$ for all $i \in [n]$, $j \in [w_i - 1]$. The cost remains the same. Any solution for the new set of constraints clearly satisfies the original one (and with the same cost).

If all the basic constraints of $\mathcal{F}$ are of the form $x_1 \bigvee \cdots \bigvee x_k$ (i.e. if all constraints are *monotone* functions) then we proceed as follows. For any variable x_i of weight w_i we introduce w_i new variables $y_i^1, \ldots, y_i^{w_i}$. Any constraint $f(x_1, ..., x_k)$ is substituted by the $w_1 w_2 \cdots w_k$ constraints

$$\{f(y_1^{j_1}, ..., y_k^{j_k}) : j_i \in [w_1], \ldots, j_k \in [w_k]\} .$$

It is not difficult to verify that if we have a feasible assignment for the new problem such that, for some i, j, $y_i^j = 0$, then we can set $y_i^h = 0$ for all $h \in [w_i]$ without contradicting any constraint. Since no 0 is changed to a 1, a solution for the non-weighted instance can be converted into a solution for the weighted instance without increasing the cost. $\Box$

3.3 Bases and First Reductions

In this subsection we set up some preliminary results that will play a role in the presentation of our results. First, we develop some shorthand notation for the constraint families: (1) $\mathcal{F}_0$ (respectively, $\mathcal{F}_1$) is the family of 0-valid (respectively, 1-valid) functions; (2) $\mathcal{F}_{\mathrm{2M}}$ is the family of 2-monotone functions; (3) $\mathcal{F}_{\mathrm{HS}}$ is the family of IHS-B functions; (4) $\mathcal{F}_{\mathrm{2A}}$ is the family of width-2 affine functions; (5) $\mathcal{F}_{\mathrm{2CNF}}$ is the family of 2CNF functions; (6) $\mathcal{F}_{\mathrm{A}}$ is the family of affine functions; (7) $\mathcal{F}_{\mathrm{WP}}$ is the family of weakly positive functions; (8) $\mathcal{F}_{\mathrm{WN}}$ is the family of weakly negative functions.

Definition 16 (Basis) *A constraint family $\mathcal{F}'$ is a basis for a constraint family $\mathcal{F}$ if any constraint of $\mathcal{F}$ can be expressed as a conjunction of constraints drawn from $\mathcal{F}'$.*

Thus, for example, a basis for an affine constraint is the set $\mathcal{F}' \cup \mathcal{F}''$ where $\mathcal{F}' = \{x_1 \oplus x_2 ... \oplus x_p = 0 \mid p \geq 1\}$ and $\mathcal{F}'' = \{x_1 \oplus x_2 ... \oplus x_p = 1 \mid p \geq 1\}$, a basis for a width-2 affine constraint is the set $\mathcal{F} = \{x \oplus y = 0, x \oplus y = 1, x = 0, x = 1\}$, and a basis for a 2CNF constraint is the set $\mathcal{F} = \{x \bigvee y, \neg x \bigvee y, \neg x \bigvee \neg y, x, \neg x\}$.

The above definition is motivated by the fact that if $\mathcal{F}'$ is a basis for $\mathcal{F}$, then an approximation algorithms for MIN CSP($\mathcal{F}'$) (resp. MIN ONES($\mathcal{F}'$)) yields an approximation algorithm for MIN CSP($\mathcal{F}$) (resp. MIN ONES ($\mathcal{F}$)). This is asserted below.

Theorem 17
If $\mathcal{F}'$ is a basis for $\mathcal{F}$, then MIN WEIGHT CSP($\mathcal{F}$) *(resp.* MIN WEIGHT ONES($\mathcal{F}$)*) is A-reducible to* MIN WEIGHT CSP($\mathcal{F}'$) *(resp.* MIN WEIGHT ONES($\mathcal{F}'$)*).*

The above theorem follows from Proposition 14 and the next two propositions.

Proposition 18 *If $f(\vec{x}) = f_1(\vec{x}) \bigwedge \cdots \bigwedge f_k(\vec{x})$, then the family $\{f_1, \ldots, f_k\}$ perfectly k-implements $\{f\}$.*

Proof: The collection $\{f_1(\vec{x}), \ldots, f_k(\vec{x})\}$ is a perfect k-implementation of $f(\vec{x})$. □

Proposition 19 *If a constraint family $\mathcal{F}'$ perfectly implements every function $f \in \mathcal{F}$, then* MIN WEIGHT ONES($\mathcal{F}$) *is AP-reducible to* MIN WEIGHT ONES($\mathcal{F}'$).

Proof: Consider an instance $\mathcal{I}$ of MIN WEIGHT ONES ($\mathcal{F}$) and substitute each constraint by a perfect implementation, thus obtaining an instance $\mathcal{I}'$ of MIN WEIGHT ONES($\mathcal{F}'$). Give weight 0 to the auxiliary variables. Each feasible solution for $\mathcal{I}$ can be extended to a feasible solution for $\mathcal{I}'$ with the same cost. Conversely, any feasible solution for $\mathcal{I}'$, when restricted to the variables of $\mathcal{I}$ is feasible for $\mathcal{I}$ and has the same cost. This is an AP-reduction. □

To simplify the presentation of algorithms, it will be useful to observe that, for a family $\mathcal{F}$, finding an approximation algorithm for MIN CSP($\mathcal{F}$) is equivalent to finding an approximation algorithm for a related family that we call $\mathcal{F}^-$.

Definition 20 *For a k-ary constraint function $f : \{0,1\}^k \to \{0,1\}$, we define $f^-(x_1, \ldots, x_k) \stackrel{\text{def}}{=} f(1 - x_1, \ldots, 1 - x_k)$. For a family $\mathcal{F} = \{f_1, \ldots, f_m\}$ we define $\mathcal{F}^- \stackrel{\text{def}}{=} \{f_1^-, \ldots, f_m^-\}$*

Proposition 21 *For every $\mathcal{F}$,* MIN WEIGHT CSP($\mathcal{F}^-$) *is A-reducible to* MIN WEIGHT CSP($\mathcal{F}$).

Proof: The reduction substitutes every constraint $f(\vec{x})$ from $\mathcal{F}$ with the constraint $f^-(\vec{x})$ from $\mathcal{F}^-$. A solution for the latter problem is converted into a solution for the former one by complementing the value of each variable. The transformation preserves the cost of the solution. □

A technical result by Khanna et al. [15] will be used extensively.

Lemma 22 ([15]) *Let $\mathcal{F}$ be a family that contains a not 0-valid and a not 1-valid function. Then*

(1) *If $\mathcal{F}$ contains a function that is not C-closed, then $\mathcal{F}$ perfectly implements the unary constraints x and $(\neg x)$.*

(2) *Otherwise, $\mathcal{F}$ perfectly implements the binary constraints $(x \oplus y = 1)$ and $(x = y)$.*

One relevant consequence (that also uses an idea from [3]) is the following.

Lemma 23 *Let $\mathcal{F}$ be a family that contains a not 0-valid and a not 1-valid function. Then* MIN WEIGHT CSP($\mathcal{F} \cup \{x, (\neg x)\}$) *is A-reducible to* MIN WEIGHT CSP($\mathcal{F}$).

Proof: If $\mathcal{F}$ contains a function that is not C-closed, then x and $(\neg x)$ can be perfectly implemented using constraints from $\mathcal{F}$, and so we are done. Otherwise, given an instance $\mathcal{I}$ of MIN WEIGHT CSP($\mathcal{F} \cup \{x, (\neg x)\}$) on variables $x_1, \ldots, x_n$ and constraints $C_1, \ldots, C_m$, we define an instance $\mathcal{I}'$ of MIN WEIGHT CSP($\mathcal{F}$) whose variables are $x_1, \ldots, x_n$ and additionally one new auxiliary variable x_F. Each constraint of the form $\neg x_i$ (resp. x_i) in $\mathcal{I}$ is replaced by a constraint $x_i = x_F$ (resp. $x_i \oplus x_F = 1$). All the other constraints are not changed. Thus $\mathcal{I}'$ also has m constraints. Given a solution $a_1, \ldots, a_n, a_F$ for $\mathcal{I}'$ which satisfies m' of these constraints, notice that the assignment $\neg a_1, \ldots, \neg a_n, \neg a_F$ also satisfies the same collection of constraints (since every function in $\mathcal{F}$ is C-closed). In one of these cases the assignment to x_F is false and then we notice that a constraint of $\mathcal{I}$ is satisfied if and only if the corresponding constraint in $\mathcal{I}'$ is satisfied. Thus every solution to $\mathcal{I}'$ can be mapped to a solution of $\mathcal{I}$ with the same objective function. □

4. Containment Results (Algorithms) for MIN CSP

In this section we show the containment results described in Theorem 11. Most results described here are simple containment results which follow easily from the notion of a "basis". The more interesting result here is a constant factor approximation algorithm for IHS-B which is presented in Lemma 25.

Lemma 24 *If $\mathcal{F} \subset \mathcal{F}'$, for some $\mathcal{F}' \in \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_{2\text{M}}\}$, then* MIN WEIGHT CSP($\mathcal{F}$) *is solvable exactly in P.*

Proof: Creignou [5] and Khanna et al. [15] show that the corresponding maximization problem is solvable exactly in P. Our lemma follows immediately (since the exact problems are interreducible). □

Lemma 25 *If $\mathcal{F} \subset \mathcal{F}_{\text{HS}}$, then* MIN WEIGHT CSP($\mathcal{F}$) $\in$ APX.

Proof: By Theorem 17 and Proposition 21 it suffices to prove the lemma for the problems MIN WEIGHT CSP(IHS-B). We will show that for every B, MIN WEIGHT CSP(IHS-B) is $B + 1$-approximable.

Given an instance $\mathcal{I}$ of MIN WEIGHT CSP(IHS-B) on variables $x_1, \ldots, x_n$ with constraints $C_1, \ldots, C_m$

with weights $w_1, \ldots, w_m$, we create a linear program on variables $y_1, \ldots, y_n$ (corresponding to the Boolean variables $x_1, \ldots, x_n$) and variables $z_1, \ldots, z_m$ (corresponding to the constraints $C_1, \ldots, C_m$). For every constraint C_j in the instance $\mathcal{I}$ we create an LP constraint as follows:

1. If $C_j = x_{i_1} \bigvee \cdots \bigvee x_{i_k}$, for $k \leq B$, we create the constraint
$$z_j + y_{i_1} + \cdots + y_{i_k} \geq 1$$
2. If $C_j = \neg x_{i_1} \bigvee x_{i_2}$, we create the constraint
$$z_j + (1 - y_{i_1}) + y_{i_2} \geq 1$$
3. If $C_j = \neg x_{i_1}$ we create the constraint
$$z_j + (1 - y_{i_1}) \geq 1$$

In addition we add the constraints $0 \leq z_j, y_i \leq 1$ for every i, j. It may be verified that any integer solution to the above described LP corresponds to an assignment to the MIN CSP problem with the variable z_j set to 1 if the constraint C_j is not satisfied. Thus the objective function for the LP is to minimize $\sum_j w_j z_j$.

Given any feasible solution vector $y_1, \ldots, y_n, z_1, \ldots, z_m$ to the LP above, we show how to obtain a 0/1 vector $y_1'', \ldots, y_n'', z_1'', \ldots, z_m''$ that is also feasible such that $\sum_j w_j z_j'' \leq (B+1) \sum_j w_j z_j$.

First we set $y_i' = \min\{1, (B+1)y_i\}$ and $z_j' = \min\{1, (B+1)z_j\}$. Observe that the vector $y_1', \ldots, y_n'$, $z_1', \ldots, z_m'$ is also feasible and gives a solution of value at most $(B+1)\sum_j w_j z_j$. We now how to get an *integral* solution whose value is at most $(B+1)\sum_j w_j z_j'$. For this part we first set $y_i'' = 1$ if $y_i' = 1$ and $z_j'' = 1$ if $z_i' = 1$. Now we remove every constraint in the LP that is made redundant. Notice in particular that every constraint of type (1) is now redundant (either z_j'' or one of the y_i'''s has already been set to 1 and hence the constraint will be satisfied by any assignment to the remaining variables). We now observe that, on the remaining variables, the LP constructed above reduces to an s-t MIN CUT LP relaxation, and therefore has an optimal integral solution. We set z_j'''s and y_i'' to such an integral and optimal solution. Notice that the so obtained solution is integral and satisfies $\sum_j w_j z_j'' \leq \sum_j w_j z_j' \leq (B+1)\sum_j w_j z_j$. □

Lemma 26 *For any family $\mathcal{F} \subset \mathcal{F}_{2A}$, $\{x \oplus y = 1, x = 1\}$ perfectly implements the family $\mathcal{F}$.*

Proof: By Proposition 18 it suffices to implement the basic width-2 affine functions: namely, the functions $x \oplus y = 1$, $x \oplus y = 0$, $x = 1$ and $x = 0$. The first and the third functions are in the target family. The function $x \oplus y = 0$ is perfectly 2-implemented by the constraints $x \oplus z_{\text{AUX}} = 1$ and $y \oplus z_{\text{AUX}} = 1$. The function $x = 0$ is implemented by the constraints $x \oplus z_{\text{AUX}} = 1$ and $z_{\text{AUX}} = 1$. □

As a consequence of the above lemma and Lemma 23, we get:

Lemma 27 *For any family $\mathcal{F} \subset \mathcal{F}_{2A}$,* MIN WEIGHT CSP$(\mathcal{F})$ *A-reduces to* MIN WEIGHT CSP$(\{x \oplus y\})$.

The following lemmas show reducibility to MIN 2CNF DELETION, NEAREST CODEWORD and MIN HORN DELETION.

Lemma 28 *For any family $\mathcal{F} \subset \mathcal{F}_{2\text{CNF}}$, the family* 2CNF *perfectly implements every function in $\mathcal{F}$.*

Proof: Again it suffices to consider the basic constraints of $\mathcal{F}$ and this is some subset of
$$\{x \bigvee y, \neg x \bigvee y, \neg x \bigvee \neg y, x, \neg x\}.$$
The family 2CNF contains all the above functions except the function $\neg x \bigvee y$ which is implemented by the constraints $\neg x \bigvee \neg z_{\text{AUX}}$ and $y \bigvee z_{\text{AUX}}$. □

Lemma 29 *For any family $\mathcal{F} \subset \mathcal{F}_{\text{A}}$, the family $\{x_1 \oplus x_2 \oplus x_3 = 0, x_1 \oplus x_2 \oplus x_3 = 1\}$ perfectly implements every function in $\mathcal{F}$.*

Proof: It suffices to show implementation of the basic affine constraints, namely, constraints of the form $x_1 \oplus x_2 ... \oplus x_p = 0$ and $x_1 \oplus x_2 ... \oplus x_q = 1$ for some $p, q \geq 1$. We focus on the former type as the implementation of the latter is analogous.

First, we observe that the constraint $x_1 \oplus x_2 = 0$ is implemented by
$$\begin{aligned} x_1 \oplus x_2 \oplus z_1 &= 0 \\ x_1 \oplus x_2 \oplus z_2 &= 0 \\ x_1 \oplus x_2 \oplus z_3 &= 0 \\ z_1 \oplus z_2 \oplus z_3 &= 0. \end{aligned}$$

Now the constraint $x_1 = 0$ can be implemented by
$$\begin{aligned} x_1 \oplus z_1 &= 0 \\ x_1 \oplus z_2 &= 0 \\ x_1 \oplus z_3 &= 0 \\ z_1 \oplus z_2 \oplus z_3 &= 0. \end{aligned}$$

The width-2 constraints in the above can be expanded as before.

Finally, the constraint $x_1 \oplus x_2 ... \oplus x_p$ for any $p > 3$ can be implemented as follows. We introduce the following set of constraints using the auxiliary variables $z_1, z_2, ..., z_{p-3}$.

$$\begin{array}{rcl} x_1 \oplus x_2 \oplus z_1 & = & 0 \\ z_1 \oplus x_3 \oplus z_2 & = & 0 \\ z_2 \oplus x_4 \oplus z_3 & = & 0 \\ \vdots & \vdots & \vdots \\ z_{p-3} \oplus x_{p-1} \oplus x_p & = & 0 \end{array}$$

□

Lemma 30 *For any family $\mathcal{F} \subseteq \mathcal{F}_{\rm WP}$, the family $\{x, \neg x, \neg x \bigvee y \bigvee z\})$ perfectly implements every function in $\mathcal{F}$.*

Proof: A k-ary weakly positive constraint (for $k \geq 2$) is either of the form $x_1 \bigvee x_2 \bigvee \ldots \bigvee x_k$ or of the form $\neg x_1 \bigvee x_2 \bigvee \cdots \bigvee x_k$. For $k = 2$, the implementation of $(x \bigvee y)$ is $\{(\neg a \bigvee x \bigvee y), a\}$, and the implementation of $(\neg x \bigvee y)$ is $\{(\neg x \bigvee y \bigvee a), \neg a\}$. For $k = 3$, the implementation of $(x \bigvee y \bigvee z)$ is $\{(a \bigvee x), (\neg a \bigvee y \bigvee z)\}$ (the constraint $(a \bigvee x)$ has in turn to be implemented with the already shown method). For $k \geq 4$, we use the textbook reduction from SAT to 3SAT (see e.g. [9, Page 49]) and we observe that when applied to k-ary weakly positive constraints it yields a perfect implementation using only 3-ary weakly positive constraints. □

5. Hardness Results (Reductions) for MIN CSP

Lemma 31 (The APX-hard Case) *If $\mathcal{F} \not\subset \mathcal{F}'$, for $\mathcal{F}' \in \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_{\rm 2M}\}$, and $\mathcal{F} \subset \mathcal{F}_{\rm HS}$ then* MIN WEIGHT CSP($\mathcal{F}$) *is* APX*-hard.*

Proof: Follows immediately from the results of [15]. □

Lemma 32 (The MIN UNCUT-hard Case) *If $\mathcal{F} \not\subset \mathcal{F}'$, for $\mathcal{F}' \in \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_{\rm 2M}, \mathcal{F}_{\rm HS}\}$, and $\mathcal{F} \subset \mathcal{F}_{\rm 2A}$ then* MIN WEIGHT CSP($\mathcal{F}$) *is* MIN UNCUT*-hard.*

Proof: It suffices to show that we can perfectly implement the constraint $x \oplus y = 1$. Consider a constraint $f \in \mathcal{F}_{\rm 2A}$ such that $f \notin \mathcal{F}_{\rm HS}$. We know that f can be expressed as a conjunction of constraints drawn from the family $\{x \oplus y = 0, x \oplus y = 1, x = 0, x = 1\}$. Notice further that all of these constraints except for the constraint $x \oplus y = 1$ are also in $\mathcal{F}_{\rm HS}$. Thus f must contain, as one of its basic primitives, the constraint $x \oplus y = 1$. Now an existential quantification over all the remaining variables in f gives us a perfect implementation of $x \oplus y = 1$. □

For the MIN 2CNF DELETION-hardness proof, we need the following two simple lemmas.

Lemma 33 *Let f be a 2CNF function which is not width-2 affine. Then f can perfectly implement some function in the family $\mathcal{F} = \{(x \bigvee y), (x \bigvee \neg y), (\neg x \bigvee \neg y)\}$.*

Proof: Let f be a 2CNF function on the variables $x_1, \ldots, x_k$. f is a conjunction of constraints of the form $x_i \Rightarrow x_j$, $x_i \Rightarrow \neg x_j$ and $\neg x_i \Rightarrow x_j$. Consider a directed graph G_f on $2k$ vertices (one corresponding to every literal x_i or $\neg x_i$) which has a directed edge from a literal l_1 to a literal l_2, if this is a constraint imposed by f. We claim that the graph G_f must have vertices l_1 and l_2 such that there is a directed path from l_1 to l_2 but not the other way around. (If not, then f can be expressed as a conjunction of equality and inequality constraints.) Existentially quantifying over all other variables (except those involved in l_1 and l_2) we find that f implements the constraint $l_1 \Rightarrow l_2$, which is one of the constraints from $\mathcal{F}$. □

Lemma 34 *Given any function $f \in \mathcal{F} = \{(x \bigvee y), (x \bigvee \neg y), (\neg x \bigvee \neg y)\}$ and the function $(x \oplus y) = 1$, we can perfectly implement all the functions in $\mathcal{F}$.*

Lemma 35 (The MIN 2CNF DELETION-hard Case) *If $\mathcal{F} \not\subset \mathcal{F}'$, for $\mathcal{F}' \in \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_{\rm 2M}, \mathcal{F}_{\rm HS}, \mathcal{F}_{\rm 2A}\}$, and $\mathcal{F} \subset \mathcal{F}_{\rm 2CNF}$ then* MIN WEIGHT CSP($\mathcal{F}$) *is* MIN 2CNF DELETION*-hard.*

Proof: We need to show that we can perfectly implement the constraints $x \bigvee y$ and $\neg x \bigvee \neg y$. Since $\mathcal{F} \not\subset \mathcal{F}_{\rm HS}$, it must contain a constraint f which is not a IHS-B^+ constraint and a constraint g which is not a IHS-B^- constraint. Since both f and g are 2CNF constraints, it means that f must have $(\neg x \bigvee \neg y)$ as a basic constraint and g must have $(x \bigvee y)$ as a basic constraint in their respective maxterm representations. Observe that the maxterm representations of neither f nor g can have the basic constraints $(x \bigvee \neg y)$ and $(\neg x \bigvee y)$. Using this observation we may conclude that an existential quantification over all variables besides x, y in f will either perfectly implement the constraint $\neg x \bigvee \neg y$ or the constraint $x \oplus y = 1$. Similarly, g can perfectly implement either the constraint $x \bigvee y$ or $x \oplus y = 1$. If we get both $x \bigvee y$ and $\neg x \bigvee \neg y$, we are done. Otherwise, we have a perfect implementation of the function $(x \oplus y = 1)$. Since $\mathcal{F} \not\subseteq \mathcal{F}_{\rm 2A}$, there must exist a constraint $h \in \mathcal{F}$ which is not width-2 affine. Using Lemmas 33 and 34, we can now conclude a perfect implementation of the desired constraints. □

Lemma 36 *If $\mathcal{F} \subset \mathcal{F}_{\rm A}$ but $\mathcal{F} \not\subset \mathcal{F}'$ for any $\mathcal{F}' \in \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_{\rm 2M}, \mathcal{F}_{\rm HS}, \mathcal{F}_{\rm 2A}\}$, then* MIN WEIGHT CSP($\mathcal{F}$) *is* NEAREST CODEWORD*-hard.*

Proof: Khanna et al. [15] show that in this case $\mathcal{F}$ perfectly implements the constraint $x_1 \oplus \cdots \oplus x_p = b$ for some $p \geq 3$ and some $b \in \{0, 1\}$. Thus the family $\mathcal{F} \cup \{T, F\}$ implements the functions $x \oplus y \oplus z = 0, x \oplus y \oplus z = 1$. Thus NEAREST CODEWORD = MIN CSP($\{x \oplus y \oplus z = 0, x \oplus y \oplus z = 1\}$ is A-reducible to MIN WEIGHT CSP($\mathcal{F} \cup \{F, T\}$). Since $\mathcal{F}$ is neither

0-valid nor 1-valid, we can use Lemma 23 to conclude that MIN WEIGHT CSP($\mathcal{F}$) is NEAREST CODEWORD-hard. $\square$

Lemma 37 ([1]) NEAREST CODEWORD *is hard to approximate to within a factor of* $2^{\log^{1-\epsilon} n}$.

Proof: The required hardness of the nearest codeword problem is shown by Arora et al. [1]. The nearest codeword problem, as defined in Arora et al., works with the following problem: Given an $n \times m$ matrix A and an m-dimensional vector b, find an n-dimensional vector x which minimizes the Hamming distance between Ax and b. Thus this problem can be expressed as a MIN CSP problem with m affine constraints over n-variables. The only technical point to be noted is that these constraints have unbounded arity. In order to get rid of such long constraints, we replace a constraint of the fo rm $x_1 \oplus \cdots \oplus x_l = 0$ into $l-2$ constraints $x_1 \oplus x_2 \oplus z_1 = 0$, $z_1 \oplus x_3 \oplus z_2 = 0$, etc. on auxiliary variables $z_1, \ldots, z_{l-3}$. (The same implementation was used in Lemma 29.) This increases the number of constraints by a factor of at most n, but doe s not change the objective function. $\square$

It remains to see the MIN HORN DELETION-hard case. We will have to draw some non-trivial consequences from the fact that a family is not IHS-B.

Lemma 38 *Assume* $\mathcal{F} \not\subset \mathcal{F}_{\rm HS}$ *and either* $\mathcal{F} \subset \mathcal{F}_{\rm WP}$ *or* $\mathcal{F} \subset \mathcal{F}_{\rm WN}$. *Then* $\mathcal{F}$ *contains a non C-closed function.*

Proof: Follows from the fact that a C-closed weakly positive function is also weakly negative. $\square$

Lemma 39 *If f is a weakly positive function not expressible as IHS-B^+, then $\{f, x, (\neg x)\}$ can perfectly implement the function $(\neg x \bigvee y \bigvee z)$.*

Proof: Since f is not IHS-B^+, any maxterm representation of f must have either a maxterm $m = (\neg x \bigvee y \bigvee z \bigvee ...)$ or a maxterm $m' = (\neg x \bigvee \neg y \bigvee ...)$. But since f is weakly positive, we must have the former scenario. We first show that f can perfectly implement the functions $x = y$ and $x \bigvee y$. To get the former, we set all literals in m, besides $\neg x$ and y, to false and existentially quantify over the rest. Since m is a maxterm, the new function f' thus obtained must either be $(\neg x \bigvee y)(x \bigvee \neg y)$ or just $(\neg x \bigvee y)$. In the former case, we are done, otherwise, $\{f'(x,y), f'(y,x)\}$ perfectly implements the $x = y$ constraint. To obtain a perfect implementation of $x \bigvee y$, a similar argument can be used by setting all literals in m besides y and z to false.

We next show how the same function f can also be used to obtain a perfect implementations of $(\neg x \bigvee y \bigvee z)$ and $(\neg x \bigvee y)$. To do so, we now set all the literals in m besides $\neg x$, y and z to false. Existentially quantifying over any other variables, we get a function f'' with a truth table as given in Figure 1.

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

Figure 1. Truth-table of the constraint f''

If $C = 0$ then restricting $x = 1$ gives the $(y \oplus z = 1)$ constraint. This contradicts the weakly positive assumption and hence $C = 1$. If $A = 1$ or $D = 1$, we get a function $(x \bigvee \neg y)$. Else $A = 0$ and $D = 0$. Now if $B = 0$, we again get $(x \bigvee \neg y)$ by existentially quantifying over z, and if $B = 1$, we get the complement of 1-in-3 sat. The complement of 1-in-3 sat function along with $x = y$ can once again implement $(x \bigvee \neg y)$— simply set $x = z$. Thus we have a perfect implementation of $(x \bigvee \neg y)$.

Now using the fact that we have the function $(x \bigvee \neg y)$, we can implement $(\neg x \bigvee y \bigvee z)$ by the following collection of constraints:

$$\{f''(x,a,b), (\neg a \bigvee y), (\neg b \bigvee z)\}$$

This completes the proof. $\square$

Lemma 40 (The MIN HORN DELETION**-hard Case)** *If* $\mathcal{F} \not\subset \mathcal{F}'$, *for* $\mathcal{F}' \in \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_{\rm 2M}, \mathcal{F}_{\rm HS}, \mathcal{F}_{\rm 2A}, \mathcal{F}_{\rm 2CNF}\}$, *and either* $\mathcal{F} \subset \mathcal{F}_{\rm WP}$ *or* $\mathcal{F} \subset \mathcal{F}_{\rm WN}$, *then* MIN WEIGHT CSP($\mathcal{F}$) *is* MIN HORN DELETION-*hard.*

Proof: From the above lemmas and from Lemma 22 we have that MIN WEIGHT CSP($\{x, \neg x, \neg x \bigvee y \bigvee z\}$) is A-reducible to MIN WEIGHT CSP($\mathcal{F}$). $\square$

Lemma 41 MIN HORN DELETION *is hard to approximate to within* $2^{\log^{1-\epsilon} n}$.

Proof: Reduction from the MIN TOTAL LABEL-COVER problem. Let (q_1, q_2, V) be an instance of MIN TOTAL LABEL-COVER, where $q_1 : [R] \to [Q_1]$, $q_2 : [R] \to [Q_2]$ and $V : [R] \times [A_1] \times [A_2] \to \{0,1\}$. For any $r \in [R]$, we define $Acc(r) = \{(a_1, a_2) : V(r, a_1, a_2) = 1\}$.

We now describe the reduction. For any $r \in R$, $a_1 \in [A_1]$, and $a_2 \in [A_2]$ we have a variable v_{r,a_1,a_2} whose intended meaning is the value of $V(r, a_1, a_2)$. Moreover, for any $q \in Q_1$ (respectively, $q \in Q_2$) and any $a \in A_1$ (resp. $a \in A_2$) we have a variable $w_{q,a}$ (resp. $x_{q,a}$), with the intended meaning that its value is 1 if and only if $a \in p_1(q)$ (respectively, $a \in p_2(q)$). For any $w_{q,a}$ (resp. $x_{q,a}$) variable we have the weight-one constraint $\neg w_{q,a}$ (resp. $\neg x_{q,a}$.) The following constraints (each with weight $(A_1 Q_1 + A_2 Q_2)$) enforce the variables to have their intended meaning. Due to their weight, it is never convenient to contradict them.

$$\begin{array}{ll}\forall r \in [r]: & \bigvee_{(a_1,a_2)\in Acc(r)} v_{r,a_1,a_2}\\ \forall r \in [r], a_1 \in [A_1], a_2 \in [A_2]: & v_{r,a_1,a_2} \Rightarrow w_{q_1(r),a_1}\\ \forall r \in [r], a_1 \in [A_1], a_2 \in [A_2]: & v_{r,a_1,a_2} \Rightarrow x_{q_2(r),a_2}\end{array}$$

The constraints of the first kind can be perfectly implemented with $x \bigvee y \bigvee z$ and $x \bigvee y \bigvee \neg z$ (see Lemma 30). It can be checked that this is an A-reduction from MIN TOTAL LABEL-COVER to MIN HORN DELETION. □

6. MIN ONES vs. MIN CSP

We begin this section with the following easy relation between MIN CSP and MIN ONES problems.

Proposition 42
For any constraint family $\mathcal{F}$, MIN WEIGHT ONES($\mathcal{F}$) *is A-reducible to* MIN WEIGHT CSP($\mathcal{F} \cup \{\neg x\}$).

Proof: Let $\mathcal{I}$ be an instance of MIN WEIGHT ONES($\mathcal{F}$) over variables $x_1, \ldots, x_n$ with weights $w_1, \ldots, w_n$. Let $w_{\max}$ be the largest weight. We construct an instance $\mathcal{I}'$ of MIN WEIGHT CSP($\mathcal{F} \cup \{\neg x\}$) by leaving the constraints of $\mathcal{I}$ (each with weight $nw_{\max}$), and adding a constraint $\neg x_i$ of weight w_i for any $i = 1, \ldots, n$. Whenever the constraints of $\mathcal{I}$ are satisfiable, it will be always convenient to satisfy them in $\mathcal{I}'$. □

Reducing a MIN CSP problem to a MIN ONES problem is slightly less obvious.

Proposition 43

(1) *If, for any $f \in \mathcal{F}$, $\mathcal{F}'$ perfectly implements $(f(\vec{x}) \bigvee y)$, then* MIN WEIGHT CSP($\mathcal{F}$) *A-reduces to* MIN WEIGHT ONES($\mathcal{F}'$).

(2) *If, for any $f \in \mathcal{F}$, $\mathcal{F}'$ perfectly implements $(f(\vec{x}) \oplus y = 1)$, then* MIN WEIGHT CSP($\mathcal{F}$) *A-reduces to* MIN WEIGHT ONES($\mathcal{F}'$).

Proof: In both cases, we use an auxiliary variable y_j for any constraint C_j. The variable takes the same weight as the constraint. The original variables have weight zero. In the first case, a constraint C_j is replaced by (the implementation of) $C_j \bigvee y_j$; in the second case by (the implementation of) $y_j = \neg C_j$. Given an assignment for the first case, we may assume as well that the ys satisfy $y_j = \neg C_j$, since if C_j is satisfied by the assignment there is no point in having $y_j = 1$. Thus, we note that the total weight of non-zero variables in the MIN ONES instance equals the total weight of non-satisifed constraints in the MIN CSP instance. □

7. Containment Results for MIN ONES

Lemma 44 (Poly-time Solvable Cases) *If $\mathcal{F} \subseteq \mathcal{F}'$ for $\mathcal{F}' \in \{\mathcal{F}_0, \mathcal{F}_{\mathrm{WN}}, \mathcal{F}_{2\mathrm{A}}\}$, then* MIN WEIGHT ONES ($\mathcal{F}$) *is solvable exactly in polynomial time*

Proof: Follows from the results of Khanna et al. [15] and from the observation that for a family $\mathcal{F}$, solving to optimality MIN WEIGHT ONES ($\mathcal{F}$) reduces to solving to optimality MAX WEIGHT ONES($\mathcal{F}^-$). □

Lemma 45 *If $\mathcal{F} \subseteq \mathcal{F}'$ for $\mathcal{F}' \in \{\mathcal{F}_{2\mathrm{CNF}}, \mathcal{F}_{\mathrm{HS}}\}$, then* MIN WEIGHT ONES ($\mathcal{F}$) *is in* APX.

Proof: For the case $\mathcal{F} \subseteq \mathcal{F}_{2\mathrm{CNF}}$, a 2-approximate algorithm is given by Hochbaum et al. [12].

Consider now the case $\mathcal{F} \subseteq \mathcal{F}_{\mathrm{HS}}$. From Theorem 17 it is sufficient to consider only basic IHS-B constraints. Since IHS-B^- constraints are weakly negative, we will restrict to basic IHS-B^+ constraints. We use linear-programming relaxations and deterministic rounding. Let k be the maximum arity of a function in $\mathcal{F}$, we will give a k-approximate algorithm. Let $\phi = \{C_1, \ldots, C_m\}$ be an instance of MIN WEIGHT ONES ($\mathcal{F}$) over variable set $X = \{x_1, \ldots, x_n\}$ with weights $w_1, \ldots, w_n$. The following is an integer linear programming formulation of finding the minimum weight satisfying assignment for ϕ.

$$\begin{array}{lll}\text{min} & \sum_i w_i y_i & \\ \text{Subject to} & & \\ & y_{i_1} + \ldots + y_{i_h} \geq 1 & \forall (x_{i_1} \bigvee \ldots \bigvee x_{i_h}) \in \phi \\ & y_{i_1} - y_{i_2} \geq 0 & \forall (x_{i_1} \bigvee \neg x_{i_2}) \in \phi \\ & y_i = 0 & \forall \neg x_i \in \phi \\ & y_i = 1 & \forall x_i \in \phi \\ & y_i \in \{0,1\} & \forall i \in \{1, \ldots, n\}\end{array} \qquad \text{(SCB)}$$

Consider now the linear programming relaxation obtained by relaxing the $y_i \in \{0,1\}$ constrains into $0 \leq y_i \leq 1$. We first find an optimum solution $\mathbf{y}^*$ for the relaxation, and we then define a 0/1 solution by setting $y_i = 0$ if $y_i^* < 1/k$, and $y_i = 1$ if $y_i^* \geq 1/k$. It is easy to see that this rounding increases the cost of the solution at most k times and that the obtained solution is feasible for (SCB). □

Lemma 46 *For any $\mathcal{F} \subseteq \mathcal{F}_{\mathrm{A}}$,* MIN WEIGHT ONES ($\mathcal{F}$) *is A-reducible to* NEAREST CODEWORD.

Proof: From Lemma 29 and Proposition 19, we have that MIN WEIGHT ONES ($\mathcal{F}$) AP-reduces to MIN WEIGHT ONES($\{x \oplus y \oplus z = 0, x \oplus y \oplus z = 1\}$). From Proposition 42, we have that MIN WEIGHT ONES ($\mathcal{F}$) A-reduces to NEAREST CODEWORD. □

Lemma 47 *For any $\mathcal{F} \subseteq \mathcal{F}_{\mathrm{WP}}$,* MIN WEIGHT ONES ($\mathcal{F}$) *is A-reducible to* MIN HORN DELETION.

Proof: Follows from Lemma 30, Proposition 19, and Proposition 42. □

8. Hardness Results for MIN ONES

Lemma 48 (APX-hard Cases)
If $\mathcal{F}$ does not satisfy the hypothesis of Lemma 44, then MIN WEIGHT ONES $(\mathcal{F})$ *is* APX*-hard.*

Proof: This part essentially follows from the proof of [15]. The major steps are as follows: We first argue that either $\mathcal{F}$ implements some function of the form $x_1 \bigvee x_2 \bigvee \cdots \bigvee x_k$, or the functions $x_1 \oplus x_2 \oplus x_3 = 0/1$ or the function $x_1 \bigvee \neg x_2$. In the first case, we get a problem that is as hard as Vertex Cover. In the second case we get a much harder problem (NCP). In the final case we need to work some more. In this case again we show that with $\{f, x, \neg x\}$ we can implement the function $x \bigvee y$. Furthermore, we show that for any function f, MIN WEIGHT ONES$(f, x, \neg x)$ AP-reduces to MIN WEIGHT ONES$(f, x_1 \bigvee \neg x_2)$. Thus once again we are down to a function which is at least as hard as VERTEX COVER. □

From now on we will assume that $\mathcal{F}$ is not 0-valid, nor weakly negative, nor width-2 affine.

Lemma 49 *If $\mathcal{F}$ is affine but not width-2 affine nor 0-valid then* MIN WEIGHT ONES$(\{x \oplus y \oplus z = 0, x \oplus y \oplus z = 1\})$ *is AP-reducible to* MIN WEIGHT ONES $(\mathcal{F})$.

Proof: From [15] we have that $\mathcal{F}$ implements the function $x_1 \oplus \cdots \oplus x_p = b$ for some $p \geq 3$ and some $b \in \{0, 1\}$. Also the existence of non 0-valid function implies we can either (essentially) implement the function T or the function $x \oplus y = 1$. In the former case we can set the variables $x_4, \ldots, x_p$ to 1 and thus implement either the constraints $x_1 \oplus x_2 \oplus x_3 = 0$ and $x_1 \oplus x_2 = 1$ or the constraints $x_1 \oplus x_2 \oplus x_3 = 1$ and $x_1 \oplus x_2 = 0$. In the latter case, we can get rid of the variables in $x_1 \oplus \cdots \oplus x_p = p$ in pairs and thus $\mathcal{F}$ either implements the functions $x_1 \oplus x_2 \oplus x_3 = 0/1$ or it implements the functions $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0/1$.

In the first and third cases listed above we immediately implement the family $\{x \oplus y \oplus z = 0, x \oplus y \oplus z = 1\}$ and so we are done. In the second and fourth cases this will not be possible (in the second case we always have 1-valid constraint and in the last case we always have constraints of even width). So we will show how to reduce the problem MIN WEIGHT ONES$(\{x \oplus y \oplus z = 0, x \oplus y \oplus z = 1\})$ to these problems. The basic idea behind the reductions is that if we have available a variable W which we know is zero, then we can implement the constraint $x \oplus y \oplus z = 0/1$. In the second case above, we only need to implement the constraint $x \oplus y \oplus z = 0$ and this is done using the constraints $x \oplus y \oplus u_{\mathrm{AUX}} = 1$ and $u_{\mathrm{AUX}} \oplus W \oplus z = 1$. In the fourth case above, the constraint $x \oplus y \oplus z = b$ is implemented using the constraint $x \oplus y \oplus z \oplus W = b$. To create such a variable we simply introduce in every instance of the reduced problem an auxiliary variable W and place a very large weight on it, so that any small weight assignment to the variables is forced to make W a zero. □

Lemma 50 MIN WEIGHT ONES$(\{x \oplus y \oplus z = 1, x \oplus y \oplus z = 0\})$ *is* NEAREST CODEWORD*-hard and hard to approximate to within a factor of* $2^{\log^{\epsilon} n}$.

Proof: The NEAREST CODEWORD-hardness follows from Lemma 29 and Proposition 43. The hardness of approximation is due to Lemma 37. □

Lemma 51
MIN WEIGHT ONES$(\{x \bigvee y \bigvee z, x \bigvee y \bigvee \neg z, x \bigvee \neg y\})$ *is hard to approximate within* $2^{\log^{1-\epsilon} n}$ *for any* $\epsilon > 0$.

Proof: Follows from Lemma 41 and Proposition 43. □

Lemma 52 *If $\mathcal{F}$ is weakly positive and not IHS-B (nor 0-valid) then* MIN WEIGHT ONES $(\mathcal{F})$ *is* MIN HORN DELETION*-hard.*

Proof: Similar to the proof of Lemma 39. □

Lemma 53 *If $\mathcal{F}$ is not 2CNF, nor IHS-B, nor affine, nor weakly positive (nor 0-valid nor weakly negative), then* MIN ONES $(\mathcal{F})$ *is* poly-APX*-hard and* MIN WEIGHT ONES $(\mathcal{F})$ *is hard to approximate to within any factor.*

Proof: We first show how to handle the weighted case. The hardness for the unweighted case will follow easily. Consider a function $f \in \mathcal{F}$ which is not weakly positive. For such an f, there exists assignments $\vec{a}$ and $\vec{b}$ such that $f(\vec{a}) = 1$ and $f(\vec{b}) = 0$ and $\vec{a}$ is zero in every coordinate where $\vec{b}$ is zero. (Such a input pair exists for every non-monotone function f and every monotone function is also weakly positive.) Now let f' be the constraint obtained from f by restricting it to inputs where $\vec{b}$ is one, and setting all other inputs to zero. Then f' is a satisfiable function which is not 1-valid. We can now apply Schaefer's theorem [25] to conclude that SAT$(\mathcal{F} \cup \{f'\})$ is hard to decide. We now reduce an instance of deciding SAT$(\mathcal{F} \cup \{f'\})$ to approximating MIN WEIGHT CSP$(\mathcal{F})$. Given an instance $\mathcal{I}$ of SAT$(\mathcal{F} \cup \{f'\})$ we create an instance which has some auxiliary variables $W_1, \ldots, W_k$ which are all supposed to be zero. This in enforced by giving them very large weights. We now replace every occurence of the constraint f' in $\mathcal{I}$ by the constraint f on the corresponding variables with the W_i's in place which were set to zero in f to obtain f'. It is clear that if a "small" weight solution exists to the resulting MIN WEIGHT CSP problem, then $\mathcal{I}$ is satisfiable, else it is not. Thus we conclude it is NP-hard to approximate MIN WEIGHT CSP to within any bounded factors.

For the unweighted case, it suffices to observe that by using polynomially bounded weights above, we get a poly-APX hardness. Further one can get rid of weights entirely by replicating variables. □

Lemma 54 ([25]) *Let $\mathcal{F}$ be a constraint family that is not 0-valid, nor 1-valid, nor weakly positive, nor weakly negative, nor affine, nor 2CNF. Then, given a set of constraints from $\mathcal{F}$ it is* NP*-hard to decide if they are satsifiable.*

Acknowledgements

We thank the anonymous referees and Jean-Pierre Seifert for several helpful comments.

References

[1] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, pages 724–733, 1993.

[2] S. Arora and M. Sudan. Improved low degree testing and its applications. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, 1997. To appear.

[3] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results (3rd version). Technical Report TR95-24, Electronic Colloquium on Computational Complexity, 1995. Preliminary version in *Proc. of FOCS'95*.

[4] D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1993.

[5] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51(3):511–522, 1995.

[6] P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan. Structure in approximation classes. In *Proceedings of the 1st Combinatorics and Computing Conference*, pages 539–548. LNCS 959, Springer-Verlag, 1995.

[7] P. Crescenzi and A. Panconesi. Completeness in approximation classes. *Information and Computation*, 93:241–262, 1991. Preliminary version in *Proc. of FCT'89*.

[8] P. Crescenzi, R. Silvestri, and L. Trevisan. To weight or not to weight: Where is the question? In *Proceedings of the 4th IEEE Israel Symposium on Theory of Computing and Systems*, pages 68–77, 1996.

[9] M. Garey and D. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979.

[10] N. Garg, V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996. Preliminary version in *Proc. of STOC'93*.

[11] F. Gavril. Manuscript cited in [9], 1974.

[12] D. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Mathematical Programming*, 62:69–83, 1993.

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

[14] S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, 1996.

[15] S. Khanna, M. Sudan, and D. Williamson. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, 1997. To appear.

[16] P. Klein, A. Agarwal, R. Ravi, and S. Rao. Aprroximation through multicommodity flow. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 726–737, 1990.

[17] P. Klein, S. Plotkin, S. Rao, and É. Tardos. Approximation algorithms for steiner and directed multicuts. To appear on *Journal of Algorithms.*, 1996.

[18] P. Kolaitis and M. Thakur. Logical definability of NP optimization problems. *Information and Computation*, 115(2):321–353, 1994.

[19] P. Kolaitis and M. Thakur. Approximation properties of NP minimization classes. *Journal of Computer and System Sciences*, 50:391–411, 1995. Preliminary version in *Proc. of Structures91*.

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

[21] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994. Preliminary version in *Proc. of STOC'93*.

[22] G. Nemhauser and L. Trotter. Vertex packing: structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

[23] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*.

[24] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, 1997. To appear.

[25] T. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[26] L. Trevisan, G. Sorkin, M. Sudan, and D. Williamson. Gadgets, approximation, and linear programming. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 617–626, 1996.

Appendix

A. Classification Theorems of Creignou [5] and Khanna et. al. [15]

Theorem 55 (MAXCSP **Classification Theorem) [5, 15]** *For every constraint set $\mathcal{F}$, the problem* MAXCSP$(\mathcal{F})$ *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.*

Theorem 56
(MAX ONES **Classification Theorem) [15]** *For every constraint set* $\mathcal{F}$, MAX ONES$(\mathcal{F})$ *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 ONES$(\mathcal{F})$ *is in P.*

(2) *Else if* $\mathcal{F}$ *is affine then* MAX ONES$(\mathcal{F})$ *is* APX-*complete.*

(3) *Else if* $\mathcal{F}$ *is strongly* 0*-valid or weakly negative or 2CNF then* MAX ONES$(\mathcal{F})$ *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 ONES$(\mathcal{F})$ *is* NP-*hard.*