Approximating Satisfiable Satisfiability Problems

[Extended Abstract]

Luca Trevisan*

Abstract. We study the approximability of the Maximum Satisfiability Problem (Max SAT) and of the boolean k-ary Constraint Satisfaction Problem (Max kCSP) restricted to satisfiable instances. For both problems we improve on the performance ratios of known algorithms for the unrestricted case.

Our approximation for satisfiable MAX 3CSP instances is better than any possible approximation for the unrestricted version of the problem (unless P= NP). This result implies that the requirements of perfect completeness and non-adaptiveness weaken the acceptance power of PCP verifiers.

We also present the first non-trivial results about PCP classes defined in terms of free bits that collapse to P.

1 Introduction

In the Max SAT problem we are given a boolean formula in conjunctive normal form (CNF) and we are asked to find an assignment of values to the variables that satisfies the maximum number of clauses. More generally, we can assume that each clause has a non-negative weight and that we want to maximize the total weight of satisfied clauses.

Max SAT is a standard NP-hard problem and a considerable research effort has been devoted in the last two decades to the development of approximation algorithms for it. An r-approximate algorithm for Max SAT (where $0 \leq r \leq 1$) is a polynomial-time algorithm that given a formula finds an assignment that satisfies clauses of total weight at least r times the optimum.

Max SAT is also the prototypical element of a large family of optimization problems in which we are given a set of weighted *constraints* over (not necessarily boolean) variables, and we want to find an assignment of values to such variables that maximizes the total weight of satisfied constraints. Problems of this kind, called constraint satisfaction problems, are of central interest in Artificial Intelligence. Their approximability properties are of interest in Theory of Computing since they can express the class MAX SNP [23, 19] and the computation of PCP verifiers [2, 25]; complete classifications of their approximability properties, for the case of boolean variables, appear in [9, 20]. We call Max kCSP

* trevisan@cui.unige.ch. Centre Universitaire d'Informatique, Université de Genève, Rue Général-Dufour 24, CH-1211, Genève, Switzerland.

the constraint satisfaction problem where every constraint involves at most k variables.

In this paper we consider the following restriction of the problem of r-approximating MAX SAT and MAX kCSP: given a satisfiable instance of MAX SAT (resp. MAX kCSP), find in polynomial time an assignment that satisfies at a fraction r of the total weight of clauses (resp. constraints). The problem of approximating constraint satisfaction problems restricted to satisfiable instances has been considered by Petrank [24], and called *approximation problem at gap location one*. Petrank observed that MAX SAT remains MAX SNP-complete when restricted to satisfiable instances, and proved that the same is true for other problems, such as MAX 3-COLORABLE SUBGRAPH and MAX 3-DIMENSIONAL MATCHING. More recenlty, Khanna, Sudan and Williamson [20] proved that for any MAX SNP-complete constraint satisfaction problem for which deciding satisfiability is NP-hard, the restriction to satisfiable instances remains MAX SNP-complete.

In partial constrast with the results of Petrank and of Khanna et al. we prove that restricting MAX SAT and MAX kCSP to satisfiable instances makes the problems somewhat easier, since we can exploit satisfiability to develop new algorithms with improved approximation guarantees. Our result for MAX 3CSP is particularly strong, since we will present a .514-approximate algorithm for satisfiable MAX 3CSP, while .501-approximation is NP-hard for the unrestricted MAX 3CSP problem [17]. Thus, the satisfiability restriction is not sufficient to turn a MAX SNP-complete problem into a PTAS problem, but *can change the approximation threshold*.[2] Our result for GL1-MAX 3CSP can also be reworded in the PCP terminology, and yields the interesting fact that *verifiers with perfect completeness are strictly weaker than verifier with completeness* $1-\epsilon$.

In the rest of this section we describe in more details our results, partly clarifying the obscure terminology of the previous paragraph.

The Maximum Satisfiability Problem. The MAX SAT problem appears in a paper of Johnson [18] which is the first paper where the term "approximation algorithm" was introduced. Johnson proved that his algorithm was 1/2-approximate. It has been recently showed that Johnson's algorithm is indeed 2/3-approximate [8]. In the last five years, several improved approximation algorithms for MAX SAT and its restricted versions MAX 2SAT and MAX 3SAT have been developed; we summarize such previous results in Table 1. There is a corresponding history of continuous improvements in the non-approximability; we do not mention it here (the interested reader can find it in [5]), and we only recall that the best known hardness is $7/8+\epsilon$ due to Håstad [17], and it still holds when restricting to satisfiable instances with exactly three literals per clause.

[2] The approximation threshold r_A of an optimization problem A is defined as

$$r_A = \sup\{r : A \text{ admits an } r\text{-approximate algorithm } \}$$

MAX SAT	MAX 3SAT	Due to
.75	.75	[27]
.75	.75	[14]
.758	.765*	[15] (using [14])
.762*	.77*	[11] (using [14, 15])
.765	.769	[22] (using [27, 14, 15])
	.801	[26] (using [11])
.768		[1] (using [14, 15, 11, 22, 26])
.8	**.826**	**This paper** for satisfiable instances

Table 1. Evolution of the approximation factors for MAX SAT and MAX 3SAT. The factors depicted with a $*$ do not appear explicitely in the referenced papers [15, 11].

OUR RESULTS. We present a polynomial-time algorithm that, given a satisfiable MAX SAT instance, satisfies a fraction .8 of the total weight of clauses, and an algorithm that, given a satisfiable MAX 3SAT instance, satisfies a fraction .826 of the total weight of clauses.

SOURCE OF OUR IMPROVEMENT. In both cases, we show how to reduce the given instance to an instance without unit clauses. The reduction sequentially applies a series of substitutions of values to variables. The .826 approximation for MAX 3SAT then follows by adapting the analysis of [26] to the case of no unit clauses. The .8 approximation for MAX SAT involves the use of known algorithms, with a couple of small changes.

Maximum k-ary Constraint Satisfaction Problem. The approximability of the MAX kCSP problem is an algorithmic rephrasing of the accepting power of PCP verifiers that non-adaptively read k bits of the proof. The restriction to satisfiable instances of MAX kCSP corresponds to the restriction to non-adaptive PCP verifiers with *perfect completeness*.[3] The requirement of perfect completeness and non-adaptiveness appeared in the first defitions of PCP and in several improved proofs of it [3, 2, 6, 7]. Recently, adaptiveness (with perfect completeness) was used in [5], and a verifier without perfect completeness (but non-adaptive) appears in [17]. The latter result was of particular interest, because it formerly appeared that "current techniques" could only yield PCP constructions with perfect completeness. The study of which PCP classes lie in P was initiated in [5]. The best known approximation for MAX kCSP, for general k, is 2^{1-k} [25].

OUR RESULTS. We improve the approximation to $(k+1)2^{-k}$ for satisfiable instances.

[3] A verifier has perfect completeness if it accepts a correct proof with probability 1.

SOURCE OF OUR IMPROVEMENT. We use again substitutions (but of a more general kind) as a preprocessing step. The substitutions reduce the problem to an instance where any k-ary constraint has at least $k+1$ satisfying assignments, and any such assignment is consistent with the set of linear constraints. We then take a random feasible solution for the set of linear constraints, and this satisfies each constraint with probability at least $(k+1)2^{-k}$.

Maximum 3-ary Constraints Satisfaction Problem (and 3-query PCP) The PCP Theorem states that membership proofs for any NP language can be probabilistically checked by a verifier that uses logarithmic randomness, has *perfect completeness*, *soundness*[4] 1/2 and *non-adaptively* reads a constant number of bits from the proof. Since its appearence, there was interest in understanding the tightest possible formulation of the PCP Theorem, especially in terms of how low the number of query bits could be made.

It is easy to see that, with two queries, it is impossible to get perfect completeness, while with 3 it is possible (see e.g. [5]). The challening question arises of determining which is the best soundness achievable with three bits and perfect completeness. The state of the art for this question is that NP can be checked with soundness $.75+\epsilon$ [17], while this is impossible with soundness .367 [26], unless P = NP. Furthermore, it is possible to check NP with three queries, soundness $.5+\epsilon$ and completeness $1-\epsilon$ for any $\epsilon > 0$ [17]. The latter result implies that MAX 3SAT is hard to approximate within $7/8+\epsilon$, but not when restricted to satisfiable instances. A different and more complicated proof was needed to prove the $7/8+\epsilon$ hardness result also for satisfiable instances [17]. It was an open question whether soundness $.5+\epsilon$ is achievable with three queries and perfect completeness.

Satisfiable instances	Arbitrary instances	Due to
.125	.125	[23]
.299		[5]
	.25	[25]
.367	.367	[26]
.514		**This paper**

Table 2. Evolution of the approximation factors for MAX 3CSP with and without the satisfiability promise.

OUR RESULT. We show that for PCP verifiers of NP languages with three non-adaptive queries and perfect completeness, the soundness is bounded away from .5, and has to be at least .514 (unless P = NP).

[4] Roughly speaking, the *soundness* is the probability of accepting a wrong proof (see Definition 6).

SOURCE OF OUR IMPROVEMENT. We give a .514-approximate algorithm for satisfiable instances of MAX 3CSP. A preprocessing step, which is a simplification of the one used for our MAX kCSP result, reduces the instance to an instance where any constraint has at least 3 satisfying assignments and each satisfying assignment is consistent with the set of linear constraints. We then apply two algorithms and take the best solution. In one algorithm, we reduce all the constraints to 2SAT using gadgets, extending an idea of [26]. In the other algorithm we take a random solution for the set of linear constraints.

Free bits. Besides the number of query bits, there is another very important parameter of the verifier that is studied in the field of probabilistic proof-checking: the number of *free bits*. It is a *relaxation* of the notion of query bit: if a verifier queries q bits on the proof, than it uses at most f free bits, but a verifier using f free bits can red arbitrarily many bits. The interest in this parameter (implicit in [13] and explicitly introduced in [7]) lies in the fact that the "efficiency" of the reduction from PCP to MAX CLIQUE [12] depends only on the number of free bits of the verifier (indeed, it depends only on the *amortized* number of free bits, but we will not exploit the latter notion here). Since the same reduction is used to derive the best known hardness result for MIN VERTEX COVER, further improvements in the hardness of approximating MIN VERTEX COVER could be obtained by improved PCP constructions with low free bits complexity. Roughly speaking, a verifier uses f free bits if, after making its queries to the proof, there at most 2^f possible answers that make him accept (this is why f cannot be larger than the number of query bits.) This definition has been used almost always, including in Håstad's papers on MAX CLIQUE (where he used the free bit-efficient *complete test*.) One exception is [5], where an *adaptive* version of the definition of free bits is used. We also mention that the free bit parameter has almost always been used for verifiers with perfect completeness (Bellare et al. [5] also show that one can always reduce the free bit complexity by reducing the completeness.) However, the currently best hardness result for MIN VERTEX COVER is due to Håstad [17] and uses a verifier with low free bit complexity and completeness $1-\epsilon$, for any $\epsilon > 0$.

Even in the simple case of the *non-adaptive* definition and of *perfect completeness* there were basically no result about PCP classes with low free bit complexity collapsing to P. The only result was that, with perfect completeness, it is impossible to characterize NP with only 1 free bit, while $\log 3$ free bits are sufficient [5]. It has been conjectured that with $\log 3$ free bits and perfect completeness it is possible to achieve any soundness.

OUR RESULT. Under the weak (non-adaptive) definition of free bits, we prove that a verifier with perfect completeness, that uses f free bits, and whose soundness is less than $2^f/2^{2^f-1}$ can only capture P.

SOURCE OF OUR IMPROVEMENT. We adapt the previously described reductions and algorithms.

Organization of the Paper. Basic definitions on constraint satisfaction problems, PCP, and gadgets are given in Section 2. We prove a simple combinatorial

result in Section 3. We present the MAX SAT approximation algorithms in Section 4 and the MAX kCSP approximation algorithms (as well as the implications with PCP classes) in Sections 5 and 6. The free bit parameter is discussed in Section 7. Several proofs are omitted or sketched in this extended abstract. The reader is referred to the full version of this paper for more details.

2 Definitions

For an integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$. We begin with a definition of constraint satisfaction problem, that unifies the definitions of all the problems we are interested in.

Definition 1. A (*k-ary*) *constraint function* is a boolean function $f : \{0,1\}^k \to \{0,1\}$.

When it is applied to variables $x_1, \ldots, x_k$ (see the following definitions) the function f is thought of as imposing the constraint $f(x_1, \ldots, x_k) = 1$.

Definition 2. A *constraint family* $\mathcal{F}$ is a finite collection of constraint functions. The *arity* of $\mathcal{F}$ is the maximum number of arguments of the functions in $\mathcal{F}$. A *constraint* C over a variable set $x_1, \ldots, x_n$ is a pair $C = (f, (i_1, \ldots, i_k))$ where $f : \{0,1\}^k \to \{0,1\}$ is a constraint function and $i_j \in [n]$ for $j \in [k]$. The constraint C is said to be *satisfied* by an assignment $\mathbf{a} = a_1, \ldots, a_n$ to $x_1, \ldots, x_n$ if $C(a_1, \ldots, a_n) \stackrel{\text{def}}{=} f(a_{i_1}, \ldots, a_{i_k}) = 1$. We say that constraint C is *from* $\mathcal{F}$ if $f \in \mathcal{F}$.

We will sometimes write a constraint $(f, (i_1, \ldots, i_k))$ as $(f(x_{i_1}, \ldots, x_{i_k}) = 1)$.

Definition 3 (Constraint famillies). A *literal* is either a variable or the negation of a variable. We define the following constraint families:

kCSP: the set of all h-ary functions, $h \leq k$.
kCSP^i: the set of all k-ary functions with i satisfying assignments.
kSAT: the set of all functions expressible as the `or` of at most k literals.
SAT: the set of all functions expressible as the `or` of literals.

A constraint function $f(x_1, \ldots, x_k)$ is *linear* if either $f(x_1, \ldots, x_k) = x_1 \oplus \ldots \oplus x_k$ or $f(x_1, \ldots, x_k) = 1 \oplus x_1 \oplus \ldots \oplus x_k$, where $\oplus$ is the `xor` operator.

Definition 4 (Constraint satisfaction problems). For a function family $\mathcal{F}$, MAX $\mathcal{F}$ is the optimization problem whose instances consist of m weighted constraints from $\mathcal{F}$, on n variables, and whose objective is to find an assignment to the variables which maximizes the total weight of satisfied constraints.

Note that Definitions 3 and 4 give rise to the problems MAX SAT, MAX 3SAT, and MAX kCSP, that are defined in the standard way.

Given an instance φ of a constraint satisfaction problem, we denote by $LIN(\varphi)$ the set of linear constraints of φ.

GL1-MAX $\mathcal{F}$[5] is the restriction of MAX $\mathcal{F}$ to instances where all the constraints are simultaneously satisfiable.

We say that a maximization problem is r-approximable $r < 1$ if there exists a polynomial-time algorithm that, for any instance, finds a solution whose cost is at least r times the optimum (such a solution is said to be r-approximate).

We also need the definition of *gadgets*.

Definition 5 (Gadget [5]). For $\alpha \in \mathcal{R}$, a function $f : \{0,1\}^k \to \{0,1\}$, and a constraint family $\mathcal{F}$: an *α-gadget* reducing f to $\mathcal{F}$ is a finite collection of constraints C_j from $\mathcal{F}$ over *primary variables* $x_1, \ldots, x_k$ and *auxiliary variables* $y_1, \ldots, y_n$, and associated real weights $w_j \geq 0$, with the property that, for boolean assignments $\mathbf{a}$ to $x_1, \ldots, x_k$ and $\mathbf{b}$ to $y_1, \ldots, y_n$, the following are satisfied:

$$(\forall \mathbf{a} : f(\mathbf{a}) = 1)\ (\forall \mathbf{b}) : \sum_j w_j C_j(\mathbf{a}, \mathbf{b}) \leq \alpha, \tag{1}$$

$$(\forall \mathbf{a} : f(\mathbf{a}) = 1)\ (\exists \mathbf{b}) : \sum_j w_j C_j(\mathbf{a}, \mathbf{b}) = \alpha, \tag{2}$$

$$(\forall \mathbf{a} : f(\mathbf{a}) = 0)\ (\forall \mathbf{b}) : \sum_j w_j C_j(\mathbf{a}, \mathbf{b}) \leq \alpha - 1. \tag{3}$$

Gadgets can be used in approximation algorithms in the following way [26]. Assume we have a satisfiable instance of a constraint satisfaction problem, with constraints of total weight m, and there is α-gadget reducing each such constraint to 2SAT. Then we can build a 2SAT instance ψ whose optimum is αm and such that any solution of cost c for ψ has cost at least $c-(\alpha-1)m$ for the old instance.

In a more general setting, assume that, for $i = 1, \ldots, k$, we have type-i constraints of total weight w_i, and that there exists an α_i-gadget reducing type-i constraints to 2SAT. Assume also that the whole CSP instance be satisfiable. Then the optimum of the instance is $\sum_i w_i$; applying all the gadgets we have a 2SAT instance ψ whose optimum is $\sum_i \alpha_i w_i$.

Applying a β-approximate algorithm to ψ, we obtain a solution for the original instance whose cost is at least

$$\sum_i \beta\alpha_i w_i - \sum_i (\alpha_i - 1)w_i = \sum_i (\beta - (1-\beta)(\alpha_i - 1))w_i \ .$$

In the following, we will refer to such kind of reductions as the *TSSW method*. The FGW [15, 11] algorithm for MAX 2SAT is .931-approximate.

We conclude this section with the definition of PCP classes and their relation with the approximability of MAX kCSP.

[5] GL1 stands for "Gap Location 1", which is the terminology of Petrank [24].

Definition 6 (Restricted verifier). A *verifier* V for a language L is a probabilistic polynomial-time Turing machine that during its computations has oracle access to a string called *proof*. We denote by $\mathbf{ACC}[V^\pi(x)]$ the probability over its random tosses that V accepts x when accessing proof π. We also denote by $\mathbf{ACC}[V(x)]$ the maximum of $\mathbf{ACC}[V^\pi(x)]$ over all proofs π. We say that

- V *has query complexity* q (where q is an integer) if for any input x, any proof π, and any outcome of its random bits, V reads at most q bits from π;
- V *has soundness* s if, for any $x \notin L$, $\mathbf{ACC}[V(x)] \leq s$;
- V *has completeness* c if, for any $x \in L$, $\mathbf{ACC}[V(x)] \geq c$. V has *perfect completeness* if it has completeness 1.

Definition 7 (PCP classes). $L \in \mathsf{PCP}_{c,s}[\log, q]$ if L admits a verifier V with completeness c, soundness s, query complexity q, and that uses $O(\log n)$ random bits, where n is the size of the input. We say that $L \in \mathbf{na}\mathsf{PCP}_{c,s}[\log, q]$ if V, in addition, queries the q bits *non-adaptively*.

Theorem 8 [2]. *If* GL1-MAX kCSP *is* r*-approximable, then* $\mathbf{na}\mathsf{PCP}_{1,s}[\log, k] \subseteq \mathsf{P}$ *for any* $s < r$.

3 Some Applications of the Linear Algebra Method

The linear algebra method in combinatorics [4] is a collection of techniques that prove combinatorial results making use of the following well-known fact: if we have a set of n-dimensional vectors that are linearly independent, then the size of the set is at most n. In this section we will provide some definitions and prove easy bounds using linear algebra. Despite the triviality of the results, they will have powerful applications in Sections 5 and 6.

In the following, we consider vectors in $\{0,1\}^n$ and denote by $\oplus$ the bitwise exclusive-or operation between vectors.

Definition 9. A *satisfying* table for a constraint function $f : \{0,1\}^k \to \{0,1\}$ with s satisfying assignments is a $s \times k$ boolean matrix whose rows are the satisfying assignments of f.

The satisfying table is not unique since the matrix representation imposes an order to the assignments. Even if it would be more natural to represent the satisfying assignments as a set of vectors rather than a matrix, the latter representation is more suitable for combinatorial arguments, especially because we can sometimes *see it as a set of* k *vectors of length* s.

Definition 10. A collection $\mathbf{x}_1, \ldots, \mathbf{x}_m$ of elements of $\{0,1\}^n$ is k*-dependent* if there are values $a_0, \ldots, a_m \in \{0,1\}$ such that $1 \leq |\{i = 1, \ldots, m : a_i = 1\}| \leq k$ and $a_1\mathbf{x}_1 \oplus \ldots \oplus a_m\mathbf{x}_m = a_0\mathbf{1}$. A collection is *dependent* if it is k-dependent for some k. A collection is *(*k*-)independent* if it is not (k-)dependent.

More intuitively, the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ are k-independent if any `xor` of at most k of them is different from $\mathbf{0}$ and from $\mathbf{1}$.

Lemma 11. *If $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \{0,1\}^n$ are 2-independent, then $m \leq 2^{n-1} - 1$. The bound is tight.*

Proof. All the $2m+2$ vectors $\mathbf{0}, \mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{1}, (\mathbf{1} \oplus \mathbf{x}_1), \ldots, (\mathbf{1} \oplus \mathbf{x}_m)$ are distinct. Therefore $2m + 2 \leq 2^n$. We omit the proof of tightness. □

Lemma 12. *If $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \{0,1\}^n$ are independent, then $m \leq n-1$. The bound is tight.*

Proof. The $m + 1$ vectors $\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_m$ are distinct and linearly independent in the ordinary sense. Therefore $m + 1 \leq n$. We omit the proof of tightness. □

Let now f be a k-ary constraint function with s satisfying assignments, and M be a satisfying table for f. If the columns of M are 2-independent, then $k \leq 2^{s-1} - 1$, that is $s \geq 1 + \lceil \log(k+1) \rceil$, which implies $s = 2$ if $k = 1$ and $s \geq 3$ if $k \geq 2$. If the columns of M are independent, then we can draw the stronger statement $s \geq k + 1$.

4 The MAX SAT Algorithms

Lemma 13. *If* GL1-MAX SAT *(resp.* GL1-MAX 3SAT*) restricted to instances without unit clauses is r-approximable, then it is r-approximable for arbitrary instances.*

Proof (Sketch). Let φ be a generic instance of GL1-MAX SAT. We will show how to produce an instance ψ of GL1-MAX SAT with no unit clauses such that given an assignment satisfying a fraction r of the clauses of ψ we are able to find an assignment satisfying a fraction at least r of the clauses of φ. If φ has no unit clauses then we are done. Otherwise we apply the following transformation:

1. For any unit clause $(x) \in \varphi$, we substitue 1 in any occurrence of x in φ.
2. For any unit clause $(\bar{x}) \in \varphi$, we substitue 0 in any occurrence of x in φ.

The transformation preserves satisfiability, does not contradict any clause, satisfies a certain number $s \geq 0$ of clauses. An assignment that satisfies a fraction r of the clauses in the new instance (i.e. $r(m-s)$ clauses) can be extended to an assignment to the old instance that satisfies $r(m-s)+s \geq rm$ clauses. After the transformation, there can still be unit clauses (produced from the shrinking of formerly longer clauses); in this case we recurse until we are left with a formula without unit clause (the process must eventually terminate after a linear number of transformations, since each transformation step reduces the size of the input.) □

Lemma 14. *There exists a polynomial-time .826-approximate algorithm for* GL1-MAX 3SAT *without unit clauses.*

Proof (Sketch). We adapt the analysis of [26]. □

Lemma 15. *There exists a polynomial-time .8-approximate algorithm for* GL1-MAX SAT *without unit clauses.*

Proof (Sketch). We use: (i) Johnson's algorithm [18]; (ii) the FGW algorithm, extended to length-3 and lenght-4 clauses with the TSSW method, and to longer clauses with a method of [15]; (iii) we solve the 2SAT sub-instance and then we apply a method of [10]. □

The gadget for lenght-4 clauses is new, as well as the idea of combining the reduction technique of [10] with a 2SAT algorithm.

Theorem 16. *There exists a .8-approximate algorithm for* GL1-MAX SAT *and a .826-approximate algorithm for* GL1-MAX 3SAT.

5 The MAX *k*CSP Algorithm

Lemma 17. *There exists a polynomial-time algorithm that, given an instance of* MAX kCSP φ *and a set of linear constraints S, such that $(\varphi \cup S)$ is satisfiable, produces an assignment that satisfies all the constraints of S and a fraction $(k+1)/2^k$ of the constraints of φ.*

Proof. We say that the instance (φ, S) is *simplified* if, for any constraint C of φ, C is not linear, the columns of the satisfying table of C are independent, and any satisfying assignment of C is consistent with S. Observe that if $h \leq k$ is the arity of a constraint C in a simplified instance, then, by Lemma 12 C has at least $h+1$ satisfying assignments, and a random assignment satisfies it with probability at least $(h+1)/2^h \geq (k+1)/2^k$. If the instance is simplified, then we take a random feasible solution for S; it satisfies all constraints of S and, on the average, a fraction at least $(k+1)/2^k$ of the total weight of the constraints of φ. Derandomization is possible with the method of conditional expectation. If the instance is not simplified then we repeatedly apply the following procedure until we are left with a simplified instance:

1. If $\exists C \in \varphi$ that is linear, then $\varphi := \varphi - \{C\}$ and $S := S \cup \{C\}$;
2. If $\exists C \equiv (f(x_{i_1}, \ldots, x_{i_k}) = 1) \in \varphi$ the columns whose satisfying table are not independent, then C enforces a linear relation $x_{i_j} = a_0 \oplus \bigoplus_{h \neq j} a_h x_{i_h}$. Then we replace C by $(f(x_{i_1}, \ldots, x_{i_{j-1}}, a_0 \oplus \bigoplus_{h \neq j} a_h x_{i_h}, x_{i_{j+1}}, \ldots, x_{i_k}) = 1$ and we add the equation $x_{i_j} = a_0 \oplus \bigoplus_{h \neq j} a_h x_{i_h}$ to S.
3. If $\exists C \in \varphi$ one whose satisfying assignment is inconsistent with S, then we remove such satisfying assignment from the satisfying table of C.

Note that all the actions above reduce the size of φ, so we can only perform a linear number of actions. After an action is performed that transforms (φ, S) into (φ', S'), the following invariants are preserved:

1. There exists an assignment satisfying $(\varphi' \cup S')$.
2. A solution satisfying S' and a fraction r of the constraints of φ' also satisfies S and a fraction r of the constraints of φ.

□

Theorem 18. *There exists a polynomial-time $(k+1)/2^k$-approximate algorithm for* GL1-MAX kCSP.

Proof. Let φ be a satisfiable instance of MAX kCSP. Apply the algorithm of Lemma 17 to the instance $(\varphi, \emptyset)$. □

Theorem 19. *For any $q \geq 3$, for any $s < (q+1)/2^q$, $\mathbf{na}\mathsf{PCP}_{1,s}[\log, q] \subseteq \mathsf{P}$.*

The bound of Lemma 18 above is 1/2 for MAX 3CSP. We will do better with semidefinite programming.

6 The MAX 3CSP Algorithm

Lemma 20. *Assume that* GL1-MAX 3CSP *is r-approximable in instances φ such that all constraints C of φ satisfy the following conditions*

1. *the columns of a satisfying table of C are 2-independent;*
2. *either C is linear or all its satisfying assignments are consistent with $LIN(\varphi)$.*

Then GL1-MAX 3CSP *is r-approximable.*

Proof (Sketch). Given a general instance φ of GL1-MAX 3CSP, we reduce it to an instance ψ satisfying properties 1 and 2. As usual, we run a series of modification steps until the required instance is generated. Each step is as follows

1. If a constraint $C \equiv (f(x_{i_1}, \ldots, x_{i_k}) = 1)$ has a 2-dependent satisfying table, then there are indices $j, h \in [k]$ and values $a_0, a_h \in \{0,1\}$ such that $x_{i_j} = a_0 \oplus a_h x_{i_h}$ Then, we replace each occurrence of x_{i_j} by $a_0 \oplus a_h x_{i_h}$.
2. If a non-linear constraint C has an assignment that is inconsistent with $LIN(\varphi)$, then we remove such assignment from the satisfying table of C.

□

Lemma 21. GL1-MAX 3CSP *restricted to the instances of Lemma 20 is* .5145*-approximable.*

Proof. From Lemma 11, φ has no unit constraint,the 2-ary constraints can only be from 2SAT, the 3-ary constraints must have at least three satisfying assigments. Let m_2 be the total weight of 2SAT constraints, $m^{(3)}$, $m^{(4)}$, $m^{(5)}$, $m^{(6)}$, $m^{(7)}$ be the total weight of 3-ary constraints that have, respectively, 3, 4, 5, 6, and 7 satisfying assignments. We also let $m^{(4L)}$ be the total weight of 3-ary linear constraints and $m^{(4O)} = m^{(4)} - m^{(4L)}$.

We use to algorithms and take the best solution.

In the first algorithm, we simply consider a random feasible solution for $LIN(\varphi)$. On the average, the total weight of satisfied constraints is at least

$$\frac{3}{4}\, m_2 + \frac{3}{8}\, m^{(3)} + \frac{4}{8}\, m^{(4O)} + \; m^{(4L)} + \frac{5}{8}\, m^{(5)} + \frac{6}{8}\, m^{(6)} + \frac{7}{8}\, m^{(7)} \qquad (4)$$

Derandomization is possible using the method of conditional expectation.

The other algorithm uses the TSSW method and the FGW algorithm. We have to find gadgets reducing the various possible 3-ary constraints to 2SAT constraints. The new constructions (and the old ones that we use) are listed in Table 3. All the gadgets are computer-constructed using the linear programming method of [26] and are the best possible. Using the FGW algorithm with the

Source Constraint	Target Constraint	α	Due to
3SAT	2SAT	3.5	[26]
4SAT	2SAT	6	**This paper**
$3CSP^3$	2SAT	5.5	**This paper**
$3CSP^4$ not linear	2SAT	5.5	**This paper**
$3CSP^4$ linear	2SAT	11	[5]
$3CSP^5$	2SAT	8.25	**This paper**
$3CSP^6$	2SAT	5.5	**This paper**

Table 3. Gadgets used.

TSSW method and the gadgets of Table 3, we have an algorithm that satisfies constraints of total weight at least

$$.931\, m_2 + .6205\, m^{(3)} + .241\, m^{(4L)} + .6205\, m^{(4O)} + .43075\, m^{(5)} + .6205\, m^{(6)} + .7585\, m^{(7)} \qquad (5)$$

If we take the maximum of Equation (4) and (5), we have that the total weight of satisfied constraints is at least $.5145m$, where $m = m_2 + m^{(3)} + m^{(4L)} + m^{(4O)} + m^{(5)} + m^{(6)} + m^{(7)}$. □

Theorem 22. *There exists a polynomial-time .5145-approximate algorithm for* GL1-MAX 3CSP*.*

Theorem 23. $\mathbf{na}\mathsf{PCP}_{1,.514}[\log, 3] \subseteq \mathsf{P}$.

7 Free Bits

We define free bits as a property of boolean functions. There are two possible definitions.

Definition 24. A function $f : \{0,1\}^q \to \{0,1\}$ uses f *non-adaptive free bits* if it has at most 2^f satisfying assignments. It uses f *adaptive free bits* if it can be expressed by a DNF with at most 2^f terms such that any two terms are inconsistent. A PCP verifier uses f adaptive (resp. non-adaptive) free bits if for any input, and any fixed random string, its computation (which is a function of the proof) can be expressed as a boolean function that uses f adaptive (resp. non-adaptive) free bits. $\mathsf{FPCP}_{c,s}[\log, f]$ is the class of languages admitting a PCP verifier with logarithmic randomness, completeness c, soundness s, that uses f adaptive free bits. The class $\mathbf{na}\mathsf{FPCP}_{c,s}[\log, f]$ is defines analogously by using the non-adaptive free bit parameter.

Regarding recent constructions of verifiers optimized for the free bit parameter, the verifiers that use the *Complete Test* [16] are non-adaptive, while the verifier that uses the *Extended Monomial Basis Test* [5] is adaptive.

We now state some results (the first ones with $f > 1$) about $\mathbf{na}\mathsf{FPCP}$ classes that collapse to P.

Theorem 25. *The following statements hold:*

1. $\mathbf{na}\mathsf{FPCP}_{1,s}[\log, f] \subseteq \mathbf{na}\mathsf{PCP}_{1,s}[\log, 2^{2^f-1} - 1]$.
2. $\mathbf{na}\mathsf{FPCP}_{1,s}[\log, f] \subseteq \mathsf{P}$ *for all* $s > (2^f)/2^{2^f-1}$ *and* $f \geq \log 3$.

Acknowledgements

I thank Greg Sorkin and Madhu Sudan for having checked some of the gadget constructions of this paper. I am grateful to Pierluigi Crescenzi and, again, to Madhu for helpful discussions on free bits.

References

1. G. Andersson and L. Engebretsen. Better approximation algorithms and tighter analysis for SET SPLITTING and NOT-ALL-EQUAL SAT. Manuscript, 1997.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. of FOCS'92*, pages 14–23.
3. S. Arora and S. Safra. Probabilistic checking of proofs; a new characterization of NP. In *Proc. of FOCS'92*, pages 2–13, 1992.
4. L. Babai and P. Frankl. *Linear Algebraic Methods in Combinatorics (2nd Preliminary version)*. Monograph in preparation, 1992.
5. M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results (4th version). Technical Report TR95-24, ECCC, 1996. Preliminary version in *Proc. of FOCS'95*.

6. M Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proc. of STOC'94*, pages 294–304.
7. M. Bellare and M. Sudan. Improved non-approximability results. In *Proc. of STOC'94*, pages 184–193.
8. J. Chen, D. Friesen, and H. Zheng. Tight bound on Johnson's algorithm for MaxSAT. In *Proc. of CCC'97*. To appear.
9. N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *JCSS*, 51(3):511–522, 1995.
10. P. Crescenzi and L. Trevisan. MAX NP-completeness made easy. Manuscript, 1996.
11. U. Feige and M. Goemans. Approximating the value of two provers proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proc. of ISTCS'95*, pages 182–189.
12. U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996. Also *Proc. of FOCS91*.
13. U. Feige and J. Kilian. Two prover protocols - low error at affordable rates. In *Proceedings of STOC'94*, pages 172–183.
14. M. Goemans and D. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM J. Disc. Math.*, 7(4):656–666, 1994. Also *Proc. of IPCO'93*.
15. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. Also *Proc. of STOC'94*.
16. J. Håstad. Testing of the long code and hardness for clique. In *Proc. STOC'96*, pages 11–19.
17. J. Håstad. Some optimal inapproximability results. In *Proc. STOC'97*, pages 1–10.
18. D.S. Johnson. Approximation algorithms for combinatorial problems. *JCSS*, 9:256–278, 1974.
19. S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. In *Proc. FOCS'94*, pages 819–830.
20. S. Khanna, M. Sudan, and D.P. Williamson. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In *Proc. STOC'97*, pages 11–20.
21. H.C. Lau and O. Watanabe. Randomized approximation of the constraint satisfaction problem. In *Proc. of SWAT'96*, pages 76–87.
22. T. Ono, T. Hirata, and T. Asano. Approximation algorithms for the maximum satisfiability problem. In *Proc. of SWAT'96*.
23. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *JCSS*, 43:425–440, 1991. Also *Proc. of STOC'88*.
24. E. Petrank. The hardness of approximations : Gap location. *Computational Complexity*, 4:133–157, 1994. Also *Proc. of ISTCS'93*.
25. L. Trevisan. Positive linear programming, parallel approximation, and PCP's. In *Proc. of ESA'96*, pages 62–75.
26. L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming. In *Proc. of FOCS'96*, pages 617–626.
27. M. Yannakakis. On the approximation of maximum satisfiability. *J. of Algorithms*, 17:475–502, 1994. Also *Proc. of SODA'92*.

This article was processed using the LaTeX macro package with LLNCS style