Gadgets, Approximation, and Linear Programming*

Luca Trevisan † Gregory B. Sorkin ‡
Madhu Sudan § David P. Williamson‡

October 28, 1998

Abstract

We present a linear programming-based method for finding "gadgets", i.e., combinatorial structures reducing constraints of one optimization problem to constraints of another. A key step in this method is a simple observation which limits the search space to a *finite* one. Using this new method we present a number of new, computer-constructed gadgets for several different reductions. This method also answers a question posed by Bellare, Goldreich and Sudan [2] of how to prove the optimality of gadgets: LP duality gives such proofs.

The new gadgets, when combined with recent results of Håstad [9], improve the known inapproximability results for MAX CUT and MAX DICUT, showing that approximating these problems to within factors of $16/17+\epsilon$ and $12/13+\epsilon$ respectively is NP-hard, for every $\epsilon > 0$. Prior to this work, the best known inapproximability thresholds for both problems was 71/72 [2]. Without using the gadgets from this paper, the best possible hardness that would follow from [2, 9] is 18/19. We also use the gadgets to obtain an improved approximation algorithm for MAX 3SAT which guarantees an approximation ratio of .801. This improves upon the previous best bound (implicit from [8, 5]) of .7704.

1 Introduction

A "gadget" is a finite combinatorial structure which translates constraints of one optimization problem into a set of constraints of a second optimization problem. A classical example is in the reduction from 3SAT to MAX 2SAT, due to Garey, Johnson and Stockmeyer [6]. Given an instance of 3SAT on variables $X_1, \ldots, X_n$ and with clauses $C_1, \ldots, C_m$, the reduction creates an instance of MAX 2SAT on the original or "primary" variables $X_1, \ldots, X_n$ along with new or "auxiliary" variables $Y^1, \ldots, Y^m$. The clauses of the MAX 2SAT instance are obtained by replacing each clause of length 3 in the 3SAT instance with a "gadget", in this case a collection of ten 2SAT clauses. For example the clause $C_k = X_1 \vee X_2 \vee X_3$ would be replaced with the following ten

*An extended abstract of this paper was presented at the 37th IEEE Symposium on Foundations of Computer Science.

† DIMACS Center, Rutgers University, Piscataway, NJ 08854. lucat@dimacs.rutgers.edu. Part of this work was done while the author was at the University of Rome "La Sapienza" and visiting IBM Research.

‡ IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights NY 10598. {sorkin, dpw}@watson.ibm.com.

§ Laboratory for Computer Science, MIT, 545 Technology Square, Cambridge, MA 02139. madhu@theory.lcs.mit.edu. Part of this work was done while the author was at the IBM Thomas J. Watson Research Center.

clauses on the variables X_1, X_2, X_3 and a new auxiliary variable Y^k:

$$X_1,\ X_2,\ X_3,\ \neg X_1 \vee \neg X_2,\ \neg X_2 \vee \neg X_3,\ \neg X_3 \vee \neg X_1,$$
$$Y^k,\ X_1 \vee \neg Y^k,\ X_2 \vee \neg Y^k,\ X_3 \vee \neg Y^k.$$

The property satisfied by this gadget is that for any assignment to the primary variables, if clause C_k is satisfied, then 7 of the 10 new clauses can be satisfied by setting Y^k appropriately; otherwise only 6 of the 10 are satisfiable. (Notice that the gadget associated with each clause C_k uses its own auxiliary variable Y^k, and thus Y^k may be set independently of the values of variables not appearing in C_k's gadget.) Using this simple property of the gadget it is easy to see that the maximum number clauses satisfied in the MAX 2SAT instance by any assignment is $7m$ if and only if the instance of 3SAT is satisfiable. This was used by [6] to prove the NP-hardness of solving MAX 2SAT. We will revisit the 3SAT-to-2SAT reduction in Lemma 6.5.

Starting with the work of Karp [12], gadgets have played a fundamental role in showing the hardness of optimization problems. They are the core of any reduction between combinatorial problems, and they retain this role in the spate of new results on the non-approximability of optimization problems.

Despite their importance, the construction of gadgets has always been a "black art", with no general methods of construction known. In fact, until recently no one had even proposed a concrete definition of a gadget; Bellare, Goldreich and Sudan [2] finally did so, with a view to quantifying the role of gadgets in non-approximability results. Their definition is accompanied by a seemingly natural "cost" measure for a gadget. The more "costly" the gadget, the weaker the reduction. However, firstly, finding a gadget for a given reduction remained an *ad hoc* task. Secondly, it remained hard to prove that a gadget's cost was optimal.

This paper addresses these two issues. We show that for a large class of reductions, the space of potential gadgets that need to be considered is actually *finite*. This is not entirely trivial, and the proof depends on properties of the problem that is being reduced to. However, the method is very general, and encompasses a large number of problems. An immediate consequence of the finiteness of the space is the existence of a search procedure to find an optimal gadget. But a naive search would be impracticably slow, and search-based proofs of the optimality (or the non-existence) of a gadget would be monstrously large.

Instead, we show how to express the search for a gadget as a linear program (LP) whose constraints guarantee that the potential gadget is indeed valid, and whose objective function is the cost of the gadget. Central to this step is the idea of working with weighted versions of optimization problems rather than unweighted ones. (Weighted versions result in LPs, while unweighted versions would result in integer programs, IPs.) This seemingly helps only in showing hardness of weighted optimization problems, but a result due to Crescenzi, Silvestri and Trevisan [3] shows that for a large class of optimization problems (including all the ones considered in this paper), the weighted versions are exactly as hard with respect to approximation as the unweighted ones. Therefore, working with a weighted version is as good as working with an unweighted one.

The LP representation has many benefits. First, we are able to search for much more complicated gadgets than is feasible manually. Second, we can use the theory of LP duality to present short(er) proofs of optimality of gadgets and non-existence of gadgets. Last, we can solve relaxed or constrained versions of the LP to obtain upper and lower bounds on the cost of a gadget, which can be significantly quicker than solving the actual LP. Being careful in the relaxing/constraining process (and with a bit of luck) we can often get the bounds to match, thereby producing optimal gadgets with even greater efficiency!

Armed with this tool for finding gadgets (and an RS/6000, OSL, and often APL2[1]), we examine some of the known gadgets and construct many new ones. (In what follows we often talk of "gadgets reducing problem X to problem Y" when we mean "gadgets used to construct a reduction from problem X to problem Y".) Bellare et al. [2] presented gadgets reducing the computation of a "verifier" for a PCP (probabilistically checkable proof system) to several problems, including MAX 3SAT, MAX 2SAT, and MAX CUT. We examine these in turn and show that the gadgets in [2] for MAX 3SAT and MAX 2SAT are optimal, but their MAX CUT gadget is not. We improve on the efficiency of the last, thereby improving on the factor to which approximating MAX CUT can be shown to be NP-hard. We also construct a new gadget for the MAX DICUT problem, thereby strengthening the known bound on its hardness. Plugging our gadget into the reduction (specifically Lemma 4.15) of [2], shows that approximating MAX CUT to within a factor of 60/61 is NP-hard, as is approximating MAX DICUT to within a factor of 44/45.[2] For both problems, the hardness factor proved in [2] was 71/72. The PCP machinery of [2] has since been improved by Håstad [9]. Our gadgets and Håstad's result show that, for every $\epsilon > 0$, approximating MAX CUT to within a factor of $16/17 + \epsilon$ is NP-hard, as is approximating MAX DICUT to within a factor of $12/13+\epsilon$. Using Håstad's result in combination with the gadgets of [2] would have given a hardness factor of $18/19 + \epsilon$ for both problems, for every $\epsilon > 0$.

Obtaining better reductions between problems can also yield improved approximation algorithms (if the reduction goes the right way!). We illustrate this point by constructing a gadget reducing MAX 3SAT to MAX 2SAT. Using this new reduction in combination with a technique of Goemans and Williamson [7, 8] and the state-of-the-art .931-approximation algorithm for MAX 2SAT due to Feige and Goemans [5] (which improves upon the previous .878-approximation algorithm of [8]), we obtain a .801-approximation algorithm for MAX 3SAT. The best result that could be obtained previously, by combining the technique of [7, 8] and the bound of [5], was .7704. (The best previously published result is a .769-approximation algorithm, due to Ono, Hirata, and Asano [14].)

Finally, our reductions have implications for probabilistically checkable proof systems. Let $\mathsf{PCP}_{c,s}[\log, q]$ be the class of languages that admit membership proofs that can be checked by a probabilistic verifier that uses a logarithmic number of random bits, reads at most q bits of the proof, accepts correct proofs with probability at least c, and accepts strings not in the language with probability at most s. We show: first, for any $\epsilon > 0$, there exist constants c and s, $c/s > 10/9 - \epsilon$, such that $\mathsf{NP} \subseteq \mathsf{PCP}_{c,s}[\log, 2]$; and second, for all c, s with $c/s > 2.7214$, $\mathsf{PCP}_{c,s}[\log, 3] \subseteq \mathsf{P}$. The best bound for the former result obtainable from [2, 9] is $22/21 - \epsilon$; the best previous bound for the latter was 4 [16].

All the gadgets we use are computer-constructed. In the final section, we present an example of a lower bound on the performance of a gadget. The bound is not computer constructed and cannot be, by the nature of the problem. The bound still relies on defining an LP that describes the optimal gadget, and extracting the lower bound from the LP's dual.

Subsequent work. Subsequent to the original presentation of this work [17], the approximability results presented in this paper have been superseded. Karloff and Zwick [10] present a

[1]Respectively, an IBM RiscSystem/6000 workstation, the IBM Optimization Subroutine Library, which includes a linear programming package, and (not that we are partisan) IBM's APL2 programming language.

[2]Approximation ratios in this paper for maximization problems are less than 1, and represent the weight of the solution achievable by a polynomial time algorithm, divided by the weight of the optimal solution. This matches the convention used in [18, 7, 8, 5] and is the reciprocal of the measure used in [2].

7/8-approximation algorithm for MAX 3SAT. This result is tight unless NP=P [9]. The containment result $\mathsf{PCP}_{c,s}[\log, 3] \subseteq \mathsf{P}$ has also been improved by Zwick [19] and shown to hold for any $c/s \geq 2$. This result is also tight, again by [9]. Finally, the gadget construction methods of this paper have found at least two more applications. Håstad [9] and Zwick [19] use gadgets constructed by these techniques to show hardness results for two problems they consider, MAX 2LIN and MAX NAE3SAT respectively.

Version. An extended abstract of this paper appeared as [17]. This version corrects some errors, pointed out by Karloff and Zwick [11], from the extended abstract. This version also presents inapproximability results resting on the improved PCP constructions of Håstad [9], while mentioning the results that could be obtained otherwise.

Organization of this paper. The next section introduces precise definitions which formalize the preceding outline. Section 3 presents the finiteness proof and the LP-based search strategy. Section 4 contains negative (non-approximability) results and the gadgets used to derive them. Section 5 briefly describes our computer system for generating gadgets. Section 6 presents the positive result for approximating MAX 3SAT. Section 7 presents proofs of optimality of the gadgets for some problems and lower bounds on the costs of others. It includes a mix of computer-generated and hand-generated lower bounds.

2 Definitions

We begin with some definitions we will need before giving the definition of a gadget from [2]. In what follows, for any positive integer n, let $[n]$ denote the set $\{1, \ldots, n\}$.

Definition 2.1 A **(k-ary) constraint function** is a boolean function $f : \{0,1\}^k \to \{0,1\}$.

We refer to k as the arity of a k-ary constraint function f. 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 2.2 A **constraint family** $\mathcal{F}$ is a collection of constraint functions. The **arity** of $\mathcal{F}$ is the maximum of the arity of the constraint functions in $\mathcal{F}$.

Definition 2.3 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 $i_1, \ldots, i_k$ are distinct members of $[n]$. The constraint C is said to be **satisfied** by an assignment $\vec{a} = a_1, \ldots, a_n$ 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 $f \in \mathcal{F}$.

Constraint functions, constraint families and constraints are of interest due to their defining role in a variety of NP optimization problems.

Definition 2.4 For a finitely specified constraint family $\mathcal{F}$, **MAX $\mathcal{F}$** is the optimization problem given by:
INPUT: An instance consisting of m constraints $C_1, \ldots, C_m$, on n Boolean variables $X_1, \ldots, X_n$, with non-negative real weights $w_1, \ldots, w_m$. (An instance is thus a triple $(\vec{X}, \vec{C}, \vec{w})$.)
GOAL: Find an assignment $\vec{b}$ to the variables $\vec{X}$ which maximizes the weight $\sum_{j=1}^{m} w_j C_j(\vec{b})$ of satisfied clauses.

Constraint functions, families and the class MAX $\mathcal{F}$ allow a description of optimization problems and reductions in a uniform manner. For example, if $\mathcal{F}$ = 2SAT is the constraint family consisting of all constraint functions of arity at most 2 that can be expressed as the disjunction of up to 2 literals, then MAX 2SAT is the corresponding MAX $\mathcal{F}$ problem. Similarly MAX 3SAT is the MAX $\mathcal{F}$ problem defined using the constraint family $\mathcal{F}$ = 3SAT consisting of all constraint functions of arity up to 3 that can be expressed as the disjunction of up to 3 literals.

One of the motivations for this work is to understand the "approximability" of many central optimization problems that can be expressed as MAX $\mathcal{F}$ problems, including MAX 2SAT and MAX 3SAT. For $\beta \in [0,1]$, an algorithm $\mathcal{A}$ is said to be a β-approximation algorithm for the MAX $\mathcal{F}$ problem, if on every instance $(\vec{X}, \vec{C}, \vec{w})$ of MAX $\mathcal{F}$ with n variables and m constraints, $\mathcal{A}$ outputs an assignment $\vec{a}$ s.t. $\sum_{j=1}^{m} w_j C_j(\vec{a}) \geq \beta \max_{\vec{b}}\{\sum_{j=1}^{m} w_j C_j(\vec{b})\}$. We say that the problem MAX $\mathcal{F}$ is β-approximable if there exists a *polynomial time*-bounded algorithm $\mathcal{A}$ that is a β-approximation algorithm for MAX $\mathcal{F}$. We say that MAX $\mathcal{F}$ is hard to approximate to within a factor β (β-inapproximable), if the existence of a polynomial time β-approximation algorithm for MAX $\mathcal{F}$ implies NP=P.

Recent research has yielded a number of new approximability results for several MAX $\mathcal{F}$ problems (cf. [7, 8]) and a number of new results yielding hardness of approximations (cf. [2, 9]). One of our goals is to construct efficient reductions between MAX $\mathcal{F}$ problems that allow us to translate "approximability" and "inapproximability" results. As we saw in the opening example such reductions may be constructed by constructing "gadgets" reducing one constraint family to another. More specifically, the example shows how a reduction from 3SAT to 2SAT results from the availability, for every constraint function f in the family 3SAT, of a gadget reducing f to the family 2SAT. This notion of a gadget reducing a constraint function f to a constraint family $\mathcal{F}$ is formalized in the following definition.

Definition 2.5 [Gadget [2]] For $\alpha \in \mathcal{R}^+$, a constraint function $f : \{0,1\}^k \rightarrow \{0,1\}$, and a constraint family $\mathcal{F}$: an **α-gadget** (or "gadget with performance α") reducing f to $\mathcal{F}$ is a set of variables $Y_1, \ldots, Y_n$, a finite collection of real weights $w_j \geq 0$, and associated constraints C_j from $\mathcal{F}$ over **primary variables** $X_1, \ldots, X_k$ and **auxiliary variables** $Y_1, \ldots, Y_n$, with the property that, for boolean assignments $\vec{a}$ to $X_1, \ldots, X_k$ and $\vec{b}$ to $Y_1, \ldots, Y_n$, the following are satisfied:

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

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

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

The gadget is **strict** if, in addition,

$$(\forall \vec{a} : f(\vec{a}) = 0)\,(\exists \vec{b}) : \sum_j w_j C_j(\vec{a}, \vec{b}) = \alpha - 1. \tag{4}$$

We use the shorthand notation $\Gamma = (\vec{Y}, \vec{C}, \vec{w})$ to denote the gadget described above.

It is straightforward to verify that the introductory example yields a strict 7-gadget reducing the constraint function $f(X_1, X_2, X_3) = X_1 \vee X_2 \vee X_3$ to the family 2SAT.

Observe that an α-gadget $\Gamma = (\vec{Y}, \vec{C}, \vec{w})$ can be converted into an $\alpha' > \alpha$ gadget by "rescaling", i.e., multiplying every entry of the weight vector $\vec{w}$ by α'/α (although strictness is not preserved). This indicates that a "strong" gadget is one with a small α; in the extreme, a 1-gadget would be the "optimal" gadget. This intuition will be confirmed in the role played by gadgets in the construction of reductions. Before describing this, we first list the constraints and constraint families that are of interest to us.

For convenience we now give a comprehensive list of all the constraints and constraint families used in this paper.

Definition 2.6

- **Parity check** (PC) is the constraint family $\{\mathrm{PC}_0, \mathrm{PC}_1\}$, where, for $i \in \{0,1\}$, PC_i is defined as follows:

$$\mathrm{PC}_i(a,b,c) = \begin{cases} 1 & \text{if } a \oplus b \oplus c = i \\ 0 & \text{otherwise.} \end{cases}$$

Henceforth we will simply use terms such as MAX PC to denote the optimization problem MAX $\mathcal{F}$ where $\mathcal{F} = \mathrm{PC}$. MAX PC (referred to as MAX 3LIN in [9]) is the source of all our inapproximability results.

- For any $k \geq 1$, **Exactly-k-SAT** (EkSAT) is the constraint family $\{f : \{0,1\}^k \to \{0,1\} : |\{\vec{a} : f(\vec{a}) = 0\}| = 1\}$, that is, the set of k-ary **disjunctive** constraints.
- For any $k \geq 1$, **kSAT** is the constraint family $\bigcup_{l \in [k]} \mathrm{E}l\mathrm{SAT}$.
- **SAT** is the constraint family $\bigcup_{l \geq 1} \mathrm{E}l\mathrm{SAT}$.

The problems MAX 3SAT, MAX 2SAT, and MAX SAT are by now classical optimization problems. They were considered originally in [6]; subsequently their central role in approximation was highlighted in [15]; and recently, novel approximation algorithms were developed in [7, 8, 5]. The associated families are typically the targets of gadget constructions in this paper. Shortly, we will describe a lemma which connects the inapproximability of MAX $\mathcal{F}$ to the existence of gadgets reducing PC_0 and PC_1 to $\mathcal{F}$. This method has so far yielded in several cases tight, and in other cases the best known, inapproximability results for MAX $\mathcal{F}$ problems.

In addition to 3SAT's use as a target, its members are also used as sources; gadgets reducing members of MAX 3SAT to MAX 2SAT help give an improved MAX 3SAT approximation algorithm.

- **3-Conjunctive SAT** (3ConjSAT) is the constraint family $\{f_{000}, f_{100}, f_{110}, f_{111}\}$, where:
  1. $f_{000}(a,b,c) = a \wedge b \wedge c$.
  2. $f_{001}(a,b,c) = a \wedge b \wedge \neg c$
  3. $f_{011}(a,b,c) = a \wedge \neg b \wedge \neg c$
  4. $f_{111}(a,b,c) = \neg a \wedge \neg b \wedge \neg c$

Members of 3ConjSAT are sources in gadgets reducing them to 2SAT. These gadgets enable a better approximation algorithm for the MAX 3ConjSAT problem, which in turn sheds light on the the class $\mathsf{PCP}_{c,s}[\log, 3]$.

- **CUT**: $\{0,1\}^2 \to \{0,1\}$ is the constraint function given by $\mathrm{CUT}(a,b) = a \oplus b$. **CUT/0** is the family of constraints $\{\mathrm{CUT}, \mathrm{T}\}$, where $\mathrm{T}(a) = 0 \oplus a = a$. **CUT/1** is the family of constraints $\{\mathrm{CUT}, \mathrm{F}\}$, where $\mathrm{F}(a) = 1 \oplus a = \neg a$.

MAX CUT is again a classical optimization problem. It has attracted attention due to the recent result of Goemans and Williamson [8] providing a .878-approximation algorithm. An observation from Bellare et al. [2] shows that the approximability of MAX CUT/0, MAX CUT/1, and MAX CUT are all identical; this is also formalized in Proposition 4.1 below. Hence MAX CUT/0 becomes the target of gadget constructions in this paper, allowing us to get inapproximability results for these three problems.

- **DICUT**: $\{0,1\}^2 \to \{0,1\}$ is the constraint function given by $\mathrm{DICUT}(a,b) = \neg a \wedge b$.

MAX DICUT is another optimization problem to which the algorithmic results of [8, 5] apply. Gadgets whose target is DICUT will enable us to get inapproximability results for MAX DICUT.

- **2CSP** is the constraint family consisting of all 16 binary functions, i.e. $2\mathrm{CSP} = \{f : \{0,1\}^2 \to \{0,1\}\}$.

MAX 2CSP was considered in [5], which gives a .859-approximation algorithm; here we provide inapproximability results.

- **Respect of monomial basis check** (RMBC) is the constraint family $\{\mathrm{RMBC}_{ij} | i,j \in \{0,1\}\}$, where

$$\mathrm{RMBC}_{ij}(a,b,c,d) = \begin{cases} 1 & \text{if } a = 0 \text{ and } b = c \oplus i \\ 1 & \text{if } a = 1 \text{ and } b = d \oplus j \\ 0 & \text{otherwise.} \end{cases}$$

  RMBC_{00} may be thought of as the test $(c,d)[a] \stackrel{?}{=} b$, RMBC_{01} as the test $(c, \neg d)[a] \stackrel{?}{=} b$, RMBC_{10} as the test $(\neg c, d)[a] \stackrel{?}{=} b$ and RMBC_{11} as the test $(\neg c, \neg d)[a] \stackrel{?}{=} b$, where the notation $(v_1, \ldots, v_n)[i]$ refers to the $i+1$'st coordinate of the vector $(v_1, \ldots, v_n)$.

Our original interest in RMBC came from the work of Bellare et al. [2] which derived hardness results for MAX $\mathcal{F}$ using gadgets reducing every constraint function in PC and RMBC to $\mathcal{F}$. This work has been effectively superseded by Håstad's [9] which only requires gadgets reducing members of PC to $\mathcal{F}$. However we retain some of the discussion regarding gadgets with RMBC functions as a source, since these constructions were significantly more challenging, and some of the techniques applied to overcome the challenges may be applicable in other gadget constructions. A summary of all the gadgets we found, with their performances and lower bounds, is given in Table 1.

We now put forth a theorem, essentially from [2] (and obtainable as a generalization of its Lemmas 4.7 and 4.15), that relates the existence of gadgets with $\mathcal{F}$ as target, to the hardness of approximating MAX $\mathcal{F}$. Since we will not be using this theorem, except as a motivation for studying the family RMBC, we do not prove it here.

Theorem 2.7 For any family $\mathcal{F}$, if there exists an α_1-gadget reducing every function in PC to $\mathcal{F}$ and an α_2-gadget reducing every function in RMBC to $\mathcal{F}$, then for any $\epsilon > 0$, MAX $\mathcal{F}$ is hard to approximate to within $1 - \frac{.15}{.6\alpha_1 + .4\alpha_2} + \epsilon$.

In this paper we will use the following, stronger, result by Håstad.

source $f \longrightarrow$ target $\mathcal{F}$	previous α	our α	lower bound
3SAT $\longrightarrow$ 2SAT	7	3.5	3.5
3ConjSAT $\longrightarrow$ 2SAT(†)		4	4
PC $\longrightarrow$ 3SAT	4		4
PC $\longrightarrow$ 2SAT	11		11
PC $\longrightarrow$ 2CSP	11	5	5
$\mathrm{PC}_0 \longrightarrow$ CUT/0	10	8	8
$\mathrm{PC}_0 \longrightarrow$ DICUT		6.5	6.5
$\mathrm{PC}_1 \longrightarrow$ CUT/0	9		9
$\mathrm{PC}_1 \longrightarrow$ DICUT		6.5	6.5
RMBC $\longrightarrow$ 2CSP	11	5	5
RMBC $\longrightarrow$ 3SAT	4		4
RMBC $\longrightarrow$ 2SAT	11		11
$\mathrm{RMBC}_{00} \longrightarrow$ CUT/0	11	8	8
$\mathrm{RMBC}_{00} \longrightarrow$ DICUT		6	6
$\mathrm{RMBC}_{01} \longrightarrow$ CUT/0	12	8	8
$\mathrm{RMBC}_{01} \longrightarrow$ DICUT(‡)		6.5	6
$\mathrm{RMBC}_{10} \longrightarrow$ CUT/0	12	9	9
$\mathrm{RMBC}_{10} \longrightarrow$ DICUT(‡)		6.5	6
$\mathrm{RMBC}_{11} \longrightarrow$ CUT/0	12	9	9
$\mathrm{RMBC}_{11} \longrightarrow$ DICUT		7	7

Table 1: Except for the cases marked (†) and (‡) above, all gadgets described are provably optimal, and strict. (†) is the best possible strict gadget; there is a non-strict 3-gadget. The optimality of gadgets marked (‡) were not verified. All "previous" results quoted are interpretations of the results in [2], except the gadget reducing 3SAT to 2SAT, which is due to [6], and the gadget reducing PC to 3SAT, which is folklore.

Theorem 2.8 [9] For any family $\mathcal{F}$, if there exists an α_0-gadget reducing PC_0 to $\mathcal{F}$ and an α_1-gadget reducing PC_1 to $\mathcal{F}$, then for any $\epsilon > 0$, MAX $\mathcal{F}$ is hard to approximate to within $1 - \frac{1}{\alpha_0+\alpha_1} + \epsilon$.

Thus, using CUT/0, DICUT, 2CSP, EkSAT and kSAT as the target of gadget constructions from PC_0 and PC_1, we can show the hardness of MAX CUT, MAX DICUT, MAX 2CSP, MAX EkSAT and MAX kSAT respectively. Furthermore, minimizing the value of α in the gadgets gives better hardness results.

3 The Basic Procedure

The key aspect of making the gadget search spaces finite is to limit the number of auxiliary variables, by showing that duplicates (in a sense to be clarified) can be eliminated by means of proper *substitutions*. In general, this is possible if the target of the reduction is a "hereditary" family as defined below.

Definition 3.1 A constraint family $\mathcal{F}$ is **hereditary** if for any $f \in \mathcal{F}$ of arity k, and any two indices $i, j \in [k]$, the function f when restricted to $X_i \equiv X_j$ and considered as a function of $k-1$ variables, is identical (up to the order of the arguments) to some other function $f' \in \mathcal{F} \cup \{0,1\}$ (where 0 and 1 denote the constant functions).

Definition 3.2 A family $\mathcal{F}$ is **complementation-closed** if it is hereditary and, for any $f \in \mathcal{F}$ of arity k, and any index $i \in [k]$, the function f' given by $f'(X_1, \ldots, X_k) = f(X_1, \ldots, X_{i-1}, \neg X_i, X_{i+1}, \ldots, X_k)$ is contained in $\mathcal{F}$.

Definition 3.3 [Partial Gadget] For $\alpha \in \mathcal{R}^+$, $S \subseteq \{0,1\}^k$, a constraint function $f : \{0,1\}^k \to \{0,1\}$, and a constraint family $\mathcal{F}$: an **S-partial α-gadget** (or "S-partial gadget with performance α") reducing f to $\mathcal{F}$ is a finite collection of constraints $C_1, \ldots, C_m$ from $\mathcal{F}$ over primary variables $X_1, \ldots, X_k$ and finitely many auxiliary variables $Y_1, \ldots, Y_n$, and a collection of non-negative real weights $w_1, \ldots, w_m$, with the property that, for boolean assignments $\vec{a}$ to $X_1, \ldots, X_k$ and $\vec{b}$ to $Y_1, \ldots, Y_n$, the following are satisfied:

$$(\forall \vec{a} \in \{0,1\}^k : f(\vec{a}) = 1)\ (\forall \vec{b} \in \{0,1\}^n) : \sum_{j=1}^{m} w_j C_j(\vec{a}, \vec{b}) \quad \leq \quad \alpha, \tag{5}$$

$$(\forall \vec{a} \in S : f(\vec{a}) = 1)\ (\exists \vec{b} \in \{0,1\}^n) : \sum_{j=1}^{m} w_j C_j(\vec{a}, \vec{b}) \quad = \quad \alpha, \tag{6}$$

$$(\forall \vec{a} \in \{0,1\}^k : f(\vec{a}) = 0)\ (\forall \vec{b} \in \{0,1\}^n) : \sum_{j=1}^{m} w_j C_j(\vec{a}, \vec{b}) \quad \leq \quad \alpha - 1. \tag{7}$$

$$(\forall \vec{a} \in S : f(\vec{a}) = 0)\ (\exists \vec{b} \in \{0,1\}^n) : \sum_{j=1}^{m} w_j C_j(\vec{a}, \vec{b}) \quad = \quad \alpha - 1. \tag{8}$$

We use the shorthand notation $\Gamma = (\vec{Y}, \vec{C}, \vec{w})$ to denote the partial gadget.

The following proposition follows immediately from the definitions of a gadget and a partial gadget.

Proposition 3.4 For a constraint function $f : \{0,1\}^k \to \{0,1\}$, let $S_1 = \{\vec{a} \in \{0,1\}^k : f(\vec{a}) = 1\}$ and let $S_2 = \{0,1\}^k$. Then for every $\alpha \in \mathcal{R}^+$ and constraint family $\mathcal{F}$:

1. An S_1-partial α-gadget reducing f to $\mathcal{F}$ is an α-gadget reducing f to $\mathcal{F}$.
2. An S_2-partial α-gadget reducing f to $\mathcal{F}$ is a strict α-gadget reducing f to $\mathcal{F}$.

Definition 3.5 For $\alpha \geq 1$ and $S \subseteq \{0,1\}^k$, let $\Gamma = (\vec{Y}, \vec{C}, \vec{w})$ be an S-partial α-gadget reducing a constraint $f : \{0,1\}^k \to \{0,1\}$ to a constraint family $\mathcal{F}$. We say that the function $b : S \to \{0,1\}^n$ is a **witness** for the partial gadget, witnessing the set S, if $b(\vec{a})$ satisfies equations (6) and (8). Specifically:

$$(\forall \vec{a} \in S : f(\vec{a}) = 1) : \sum_{j=1}^{m} w_j C_j(\vec{a}, b(\vec{a})) = \alpha, \text{ and}$$

$$(\forall \vec{a} \in S : f(\vec{a}) = 0) : \sum_{j=1}^{m} w_j C_j(\vec{a}, b(\vec{a})) = \alpha - 1.$$

The witness function can also be represented as an $|S| \times (k+n)$-matrix W_b whose rows are the vectors $(\vec{a}, b(\vec{a}))$. Notice that the columns of the matrix correspond to the variables of the gadget, with the first k columns corresponding to primary variables, and the last n corresponding to auxiliary variables. In what follows we shall often prefer the matrix notation.

Definition 3.6 For a set $S \subseteq \{0,1\}^k$ let M_S be the matrix whose rows are the vectors $\vec{a} \in S$, let k'_S be the number of distinct columns in M_S, and let k''_S be the number of columns in M_S distinct up to complementation. Given a constraint f of arity k and a hereditary constraint family $\mathcal{F}$ that is *not* complementation-closed, an $(S, f, \mathcal{F})$-**canonical witness matrix** (for an S-partial gadget reducing f to $\mathcal{F}$) is the $|S| \times (2^{|S|} + k - k'_S)$ matrix W whose first k columns correspond to the k primary variables and whose remaining columns are all possible column vectors that are distinct from one another and from the columns corresponding to the primary variables. If $\mathcal{F}$ *is* complementation-closed, then a canonical witness matrix is the $|S| \times (2^{|S|-1} + k - k''_S)$ matrix W whose first k columns correspond to the k primary variables and whose remaining columns are all possible column vectors that are distinct up to complementation from one another and from the columns corresponding to the primary variables.

The following lemma is the crux of this paper and establishes that the optimal gadget reducing a constraint function f to a hereditary family $\mathcal{F}$ is finite. To motivate the lemma, we first present an example, due to Karloff and Zwick [11], showing that this need not hold if the family $\mathcal{F}$ is not hereditary. Their counterexample has $f(a) = a$ and $\mathcal{F} = \{\mathrm{PC}_1\}$. Using k auxiliary variables, $Y_1, \ldots, Y_k$, one may construct a gadget for the constraint X, using the constraints $X \oplus Y_i \oplus Y_j$, $1 \le i < j \le k$, with each constraint having the same weight. For an appropriate choice of this weight it may be verified that this yields a $(2-2/k)$-gadget for even k; thus the performance tends to 2 in the limit. On the other hand it can be shown that any gadget with k auxiliary variables has performance at most $2 - 2^{1-k}$; thus no finite gadget achieves the limit. It is clear that for this example the lack of hereditariness is critical: any hereditary family containing PC_1 would also contain f, providing a trivial 1-gadget.

To see why the hereditary property helps in general, consider an α-gadget Γ reducing f to $\mathcal{F}$, and let W be a witness matrix for Γ. Suppose two columns of W, corresponding to auxiliary variables Y_1 and Y_2 of Γ, are identical. Then we claim that Γ does not really need the variable Y_2. In every constraint containing Y_2, replace it with Y_1, to yield a new collection of weighted constraints. By the hereditary property of $\mathcal{F}$, all the resulting constraints are from $\mathcal{F}$. And, the resulting instance satisfies all the properties of an α-gadget. (The universal properties follow trivially, while the existential properties follow from the fact that in the witness matrix Y_1 and Y_2 have the same assignment.) Thus this collection of constraints forms a gadget with fewer variables and performance at least as good. The finiteness follows from the fact a witness matrix with distinct columns has a bounded number of columns. The following lemma formalizes this argument. In addition it also describes the canonical witness matrix for an optimal gadget — something that will be of use later.

Lemma 3.7 For $\alpha \ge 1$, set $S \subset \{0,1\}^k$, constraint $f : \{0,1\}^k \to \{0,1\}$ and hereditary constraint family $\mathcal{F}$, if there exists an S-partial α-gadget Γ reducing f to $\mathcal{F}$, with witness matrix W, then for any $(S, f, \mathcal{F})$-canonical witness matrix W', and some $\alpha' \le \alpha$, there exists an α'-gadget Γ' reducing f to $\mathcal{F}$, with W' as a witness matrix.

Proof: We first consider the case where $\mathcal{F}$ is not complementation-closed. Let $\Gamma = (\vec{Y}, \vec{C}, \vec{w})$ be an S-partial α-gadget reducing f to $\mathcal{F}$ and let W be a witness matrix for Γ. We create a gadget Γ' with $n' = 2^{|S|} - k'$ auxiliary variables $Y'_1, \ldots, Y'_{n'}$, one associated with each column of the matrix W' other than the first k.

With each variable Y_i of Γ we associate a variable Z such that the column corresponding to Y_i in W is the same as the column corresponding to Z in W'. Notice that Z may be one of the primary variables $X_1, \ldots, X_k$ or one of the auxiliary variables $Y'_1, \ldots, Y'_{n'}$. By definition of a canonical witness, such a column and hence variable Z does exist.

Now for every constraint C_j on variables $Y_{i_1}, \ldots, Y_{i_k}$ in Γ with weight w_j, we introduce the constraint C_j on variables $Y'_{i'_1}, \ldots, Y'_{i'_k}$ in Γ' with weight w_j where $Y'_{i'_l}$ is the variable associated with Y_{i_l}. Notice that in this process the variables involved with a constraint do not necessarily remain distinct. This is where the *hereditary property* of $\mathcal{F}$ is used to ensure that a constraint $C_j \in \mathcal{F}$, when applied to a tuple of non-distinct variables, remains a constraint in $\mathcal{F}$. In the process we may arrive at some constraints which are either always satisfied or never satisfied. For the time being, we assume that the constraints **0** and **1** are contained in $\mathcal{F}$, so this occurrence does not cause a problem. Later we show how this assumption is removed.

This completes the description of Γ'. To verify that Γ' is indeed an S-partial α-gadget, we notice that the universal constraints (conditions (5) and (7) in Definition 3.3) are trivially satisfied, since Γ' is obtained from Γ by renaming some variables and possibly identifying some others. To see that the existential constraints (conditions (6) and (8) in Definition 3.3) are satisfied, notice that the assignments to the variables $\vec{Y}$ that witness these conditions in Γ are allowable assignments to the corresponding variables in $\vec{Y}'$ and in fact this is what dictated our association of variables in $\vec{Y}$ to the variables in $\vec{Y}'$. Thus Γ' is indeed an S-partial α-gadget reducing f to $\mathcal{F}$, and, by construction, has W' as a witness matrix.

Last, we remove the assumption that Γ' must include constraints **0** and **1**. Any constraints **0** can be safely thrown out of the gadget without changing any of the parameters, since such constraints are never satisfied. On the other hand, constraints **1** do affect α. If we throw away a **1** constraint of weight w_j, this reduces the total weight of satisfied clauses in every assignment by w_j. Throwing away all such constraints reduces α by the total weight of the **1** constraints, producing a gadget of (improved) performance $\alpha' \leq \alpha$.

Finally, we describe the modifications required to handle the case where $\mathcal{F}$ is complementation-closed (in which case the definition of a canonical witness changes). Here, for each variable Y_i and its associated column of W, either there is an equal column in W', in which case we replace Y_i with the column's associated variable $Y'_{i'}$, or there is a complementary column in W', in which case we replace Y_i with the negation of the column's associated variable, $\neg Y'_{i'}$, The rest of the construction proceeds as above, and the proof of correctness is the same. $\Box$

It is an immediate consequence of Lemma 3.7 that an optimum gadget reducing a constraint function to a hereditary family does not need to use more than an explicitly bounded number of auxiliary variable.

Corollary 3.8 Let f be a constraint function of arity k with s satisfying assignments. Let $\mathcal{F}$ be a constraint family and $\alpha \geq 1$ be such that there exists an α-gadget reducing f to $\mathcal{F}$.

1. If $\mathcal{F}$ is hereditary then there exists an α'-gadget with at most $2^s - k'$ auxiliary variables reducing f to $\mathcal{F}$, where $\alpha' \leq \alpha$, and k' is the number of distinct variables among the satisfying assignments of f.

2. If $\mathcal{F}$ is complementation-closed then there exists an α'-gadget with at most $2^{s-1} - k''$ auxiliary variables reducing f to $\mathcal{F}$, for some $\alpha' \leq \alpha$, where k'' is the number of distinct variables, up to complementation, among the satisfying assignments of f.

Corollary 3.9 Let f be a constraint function of arity k. Let $\mathcal{F}$ be a constraint family and $\alpha \geq 1$ be such that there exists a strict α-gadget reducing f to $\mathcal{F}$.

1. If $\mathcal{F}$ is hereditary then there exists a strict α'-gadget with at most $2^{2^k} - k$ auxiliary variables reducing f to $\mathcal{F}$, for some $\alpha' \leq \alpha$.

2. If $\mathcal{F}$ is complementation-closed then there exists a strict α'-gadget with at most $2^{2^k-1}-k$ auxiliary variables reducing f to $\mathcal{F}$, for some $\alpha' \leq \alpha$.

We will now show how to cast the search for an optimum gadget as a linear programming problem.

Definition 3.10 For a constraint function f of arity k, constraint family $\mathcal{F}$, and $s \times (k+n)$ witness matrix M, $\mathrm{LP}(f, \mathcal{F}, M)$ is a linear program defined as follows:

- Let $C_1, \ldots, C_m$ be all the possible distinct constraints that arise from applying a constraint function from $\mathcal{F}$ to a set of $n+k$ Boolean variables. Thus for every j, $C_j : \{0,1\}^{k+n} \to \{0,1\}$. The LP variables are $w_1, \ldots, w_m$, where w_j corresponds to the weight of the constraint C_j. Additionally the LP has one more variable α.
- Let $S \subseteq \{0,1\}^k$ and $b : S \to \{0,1\}^n$ be such that $M = W_b$ (i.e., M is the witness matrix corresponding to the witness function b for the set S). The LP inequalities correspond to the definition of an S-partial gadget.

$$(\forall \vec{a} \in \{0,1\}^k : f(\vec{a}) = 1)\ (\forall \vec{b} \in \{0,1\}^n) : \sum_{j=1}^{m} w_j C_j(\vec{a}, \vec{b}) \quad \leq \quad \alpha, \tag{9}$$

$$(\forall \vec{a} \in S : f(\vec{a}) = 1) : \sum_{j=1}^{m} w_j C_j(\vec{a}, b(\vec{a})) \quad = \quad \alpha, \tag{10}$$

$$(\forall \vec{a} \in \{0,1\}^k : f(\vec{a}) = 0)\ (\forall \vec{b} \in \{0,1\}^n) : \sum_{j=1}^{m} w_j C_j(\vec{a}, \vec{b}) \quad \leq \quad \alpha - 1, \tag{11}$$

$$(\forall \vec{a} \in S : f(\vec{a}) = 0) : \sum_{j=1}^{m} w_j C_j(\vec{a}, b(\vec{a})) \quad = \quad \alpha - 1. \tag{12}$$

  Finally the LP has the inequalities $w_j \geq 0$.
- The objective of the LP is to minimize α.

Proposition 3.11 For any constraint function f of arity k, constraint family $\mathcal{F}$, and $s \times (k+n)$ witness matrix M witnessing the set $S \subseteq \{0,1\}^k$, if there exists an S-partial gadget reducing f to $\mathcal{F}$ with witness matrix M, then $\mathrm{LP}(f, \mathcal{F}, M)$ finds such a gadget with the minimum possible α.

Proof: The LP-generated gadget consists of k primary variables $X_1, \ldots, X_k$ corresponding to the first k columns of M; n auxiliary variables $Y_1, \ldots, Y_n$ corresponding to the remaining n columns of M; constraints $C_1, \ldots, C_m$ as defined in Definition 3.10; and weights $w_1, \ldots, w_m$ as returned by $\mathrm{LP}(f, \mathcal{F}, M)$. By construction the LP solution returns the minimum possible α for which an S-partial α-gadget reducing f to $\mathcal{F}$ with witness M exists. □

Theorem 3.12 [Main] Let f be a constraint function of arity k with s satisfying assignments. Let k' be the number of distinct variables of f and k'' be the number of distinct variables up to complementation. Let $\mathcal{F}$ be a hereditary constraint family with functions of arity at most l. Then:

- If there exists an α-gadget reducing f to $\mathcal{F}$, then there exists such a gadget with at most v auxiliary variables, where $v = 2^{s-1} - k''$ if $\mathcal{F}$ is complementation-closed and $v = 2^s - k'$ otherwise.

- If there exists a strict α-gadget reducing f to $\mathcal{F}$ then there exists such a gadget with at most v auxiliary variables, where $v = 2^{2^k-1} - k''$ if $\mathcal{F}$ is complementation-closed and $v = 2^{2^k} - k'$ otherwise.

Furthermore such a gadget with smallest performance can be found by solving a linear program with at most $|\mathcal{F}| \times (v+k)^l$ variables and 2^{v+k} constraints.

Remark: The sizes given above are upper bounds. In specific instances, the sizes may be much smaller. In particular, if the constraints of $\mathcal{F}$ exhibit symmetries, or are not all of the same arity, then the number of variables of the linear program will be much smaller.

Proof: By Proposition 3.11 and Lemma 3.7, we have that $\mathrm{LP}(f, \mathcal{F}, W_S)$ yields an optimal S-partial gadget if one exists. By Proposition 3.4 the setting $S = S_1 = \{\vec{a} | f(\vec{a}) = 1\}$ gives a gadget, and the setting $S = S_2 = \{0,1\}^k$ gives a strict gadget. Corollaries 3.8 and 3.9 give the required bound on the number of auxiliary variables; and the size of the LP then follows from the definition. □

To conclude this section, we mention some (obvious) facts that become relevant when searching for large gadgets. First, if $S' \subseteq S$, then the performance of an S'-partial gadget reducing f to $\mathcal{F}$ is also a lower bound on the performance of an S-partial gadget reducing f to $\mathcal{F}$. The advantage here is that the search for an S'-partial gadget may be much faster. Similarly, to get upper bounds on the performance of an S-partial gadget, one may use other witness matrices for S (rather than the canonical one); in particular ones with (many) fewer columns. This corresponds to making a choice of auxiliary variables not to be used in such a gadget.

4 Improved Negative Results

4.1 MAX CUT

We begin by showing an improved hardness result for the MAX CUT problem. It is not difficult to see that no gadget per Definition 2.5 can reduce any member of PC to CUT: for any setting of the variables which satisfies equation (2), the complementary setting has the opposite parity (so that it must be subject to inequality (3)), but the values of all the CUT constraints are unchanged, so that the gadget's value is still α, violating (3). Following [2], we use instead the fact that MAX CUT and MAX CUT/0 are equivalent with respect to approximation as shown below.

Proposition 4.1 MAX CUT is equivalent to MAX CUT/0. Specifically, given an instance $\mathcal{I}$ of either problem, we can create an an instance $\mathcal{I}'$ of the other with the same optimum and with the feature that an assignment satisfying constraints of total weight W to the latter can be transformed into an assignment satisfying constraints of the same total weight in $\mathcal{I}$.

Proof: The reduction from MAX CUT to MAX CUT/0 is trivial, since the family CUT/0 contains CUT; and thus the identity map provides the required reduction.

In the reverse direction, given an instance $(\vec{X}, \vec{C}, \vec{w})$ of MAX CUT/0 with n variables and m clauses, we create an instance $(\vec{X}', \vec{C}', \vec{w})$ of MAX CUT with $n+1$ variables and m clauses. The variables are simply the variables $\vec{X}$ with one additional variable called 0. The constraints of $\vec{C}$ are transformed as follows. If the constraint is a CUT constraint on variables X_i and X_j it is retained as is. If the constraint is $T(X_i)$ it is replaced with the constraint $\mathrm{CUT}(X_i, 0)$. Given a assignment $\vec{a}$ to the vector $\vec{X}'$, notice that its complement also satisfies the same number of constraints in $\mathcal{I}'$.

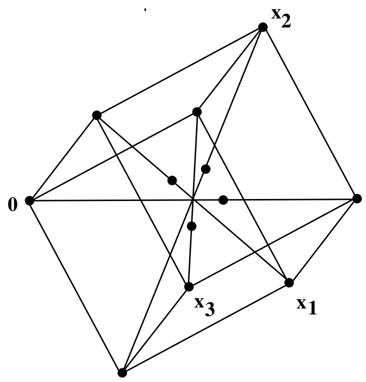

Figure 1: 8-gadget reducing PC_0 to CUT. Every edge has weight .5. The auxiliary variable which is always 0 is labelled 0.

We pick the one among the two that sets the variable 0 to 0, and then observe that the induced assignment to $\vec{X}$ satisfies the corresponding clauses of $\mathcal{I}$. □

Thus we can look for reductions to CUT/0. Notice that the CUT/0 constraint family is hereditary, since identifying the two variables in a CUT constraint yields the constant function 0. Thus by Theorem 3.12, if there is an α-gadget reducing PC_0 to CUT/0, then there is an α-gadget with at most 13 auxiliary variables (16 variables in all). Only $\binom{16}{2} = 120$ CUT constraints are possible on 16 variables. Since we only need to consider the cases when $Y_1 = 0$, we can construct a linear program as above with $2^{16-1} + 4 = 32,772$ constraints to find the optimal α-gadget reducing PC_0 to CUT/0. A linear program of the same size can similarly be constructed to find a gadget reducing PC_1 to CUT/0.

Lemma 4.2 There exists an 8-gadget reducing PC_0 to CUT/0, and it is optimal and strict.

We show the resulting gadget in Figure 1 as a graph. The primary variables are labelled x_1, x_2 and x_3, while 0 is a special variable. The unlabelled vertices are auxiliary variables. Each constraint of non-zero weight is shown as an edge. An edge between the vertex 0 and some vertex x corresponds to the constraint $T(x)$. Any other edge between x and y represents the constraint $\mathrm{CUT}(x, y)$. Note that some of the 13 possible auxiliary variables do not appear in any positive weight constraint and thus are omitted from the graph. All non-zero weight constraints have weight .5.

By the same methodology, we can prove the following.

Lemma 4.3 There exists a 9-gadget reducing PC_1 to CUT/0, and it is optimal and strict.

The gadget is similar to the previous one, but the old vertex 0 is renamed Z, and a new vertex labelled 0 is joined to Z by an edge of weight 1.

The two lemmas along with Proposition 4.1 above imply the following theorem.

Theorem 4.4 For every $\epsilon > 0$, MAX CUT is hard to approximate to within $16/17 + \epsilon$.

Proof: Combining Theorem 2.8 with Lemmas 4.2 and 4.3 we find that MAX CUT/0 is hard to approximate to within $16/17 + \epsilon$. The theorem then follows from Proposition 4.1. □

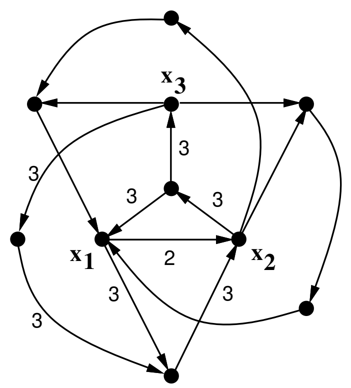

Figure 2: 8-gadget reducing PC_0 to DICUT. Edges have weight 1 except when marked otherwise.

RMBC **gadgets.** Finding RMBC gadgets was more difficult. We discuss this point since it leads to ideas that can be applied in general when finding large gadgets. Indeed, it turned out that we couldn't exactly apply the technique above to find an optimal gadget reducing, say, RMBC_{00} to CUT/0. (Recall that the $\mathrm{RMBC}_{00}(a_1, a_2, a_3, a_4)$ is the function $(a_3, a_4)[a_1] \stackrel{?}{=} a_2$.) Since there are 8 satisfying assignments to the 4 variables of the RMBC_{00} constraint, by Theorem 3.12, we would need to consider $2^8 - 4 = 252$ auxiliary variables, leading to a linear program with $2^{252} + 8$ constraints, which is somewhat beyond the capacity of current computing machines. To overcome this difficulty, we observed that for the RMBC_{00} function, the value of a_4 is irrelevant when $a_1 = 0$ and the value of a_3 is irrelevant when $a_1 = 1$. This led us to try only restricted witness functions for which $\vec{b}(0, a_2, a_3, 0) = \vec{b}(0, a_2, a_3, 1)$ and $\vec{b}(1, a_2, 0, a_4) = \vec{b}(1, a_2, 1, a_4)$ (dropping from the witness matrix columns violating the above conditions), even though it is not evident *a priori* that a gadget with a witness function of this form exists. The number of distinct variable columns that such a witness matrix can have is at most 16. Excluding auxiliary variables identical to a_1 or a_2, we considered gadgets with at most 14 auxiliary variables. We then created a linear program with $\binom{18}{2} = 153$ variables and $2^{18-1} + 8 = 131,080$ constraints. The result of the linear program was that there exists an 8-gadget with constant 0 reducing RMBC_{00} to CUT, and that it is strict. Since we used a restricted witness function, the linear program does not prove that this gadget is optimal. However, its optimality can be proved, and this will be discussed in Section 7. Similar ideas led to construction of gadgets for RMBC_{01}, RMBC_{10} and RMBC_{11} also.

4.2 MAX DICUT

As in the previous subsection, we observe that if there exists an α-gadget reducing an element of PC to DICUT, there exists an α-gadget with 13 auxiliary variables. This leads to linear programs with $16 \cdot 15$ variables (one for each possible DICUT constraint, corresponding to a directed edge) and $2^{16} + 4 = 65,540$ linear constraints. The solution to the linear programs gives the following.

Lemma 4.5 There exist 6.5-gadgets reducing PC_0 and PC_1 to DICUT, and they are optimal and strict.

The PC_0 gadget is shown in Figure 2. Again x_1, x_2 and x_3 refer to the primary variable and an edge from x to y represents the constraint $\neg x \wedge b$. The PC_1 gadget is similar, but has all edges reversed.

Theorem 4.6 For every $\epsilon > 0$, MAX DICUT is hard to approximate to within $12/13 + \epsilon$.

RMBC **gadgets.** Using the idea of a restricted witness function as in the reduction from RMBC to MAX CUT, we set up a linear program to find α-gadgets reducing elements of RMBC to DICUT given the restricted witness function. Each linear program has $18 \cdot 17 = 306$ variables and $2^{18} + 8 = 262,152$ constraints. We found that there exists a 6-gadget reducing RMBC_{00} to DICUT, and it is strict; there exist 6.5-gadgets reducing RMBC_{01} and RMBC_{10} to DICUT, and they are strict; there exists a 7-gadget reducing RMBC_{11} to DICUT, and it is strict. Once again, because we consider only restricted witness functions, the linear program does not necessarily produce an optimal gadget. In Section 7 we prove, for instance, that no better gadget exists reducing RMBC_{11} to DICUT.

4.3 MAX 2-CSP

For reducing an element of PC to the 2CSP family we need consider only 4 auxiliary variables, for a total of 7 variables. There are two non-constant functions on a single variable, and twelve non-constant functions on pairs of variables, so that there are $2 \cdot 7 + 12 \cdot \binom{7}{2} = 266$ functions to consider overall. We can again set up a linear program with a variable per function and $2^7 + 4 = 132$ linear constraints. We obtain the following.

Lemma 4.7 There exist 5-gadgets reducing PC_0 and PC_1 to 2CSP, and they are optimal and strict.

The gadget reducing PC_0 to 2CSP is the following:

$$\begin{array}{rrrr} X_1 \wedge \neg Y_1, & X_1 \wedge Y_2, & \neg X_1 \wedge Y_3, & \neg X_1 \wedge Y_4, \\ X_2 \wedge \neg Y_1, & \neg X_2 \wedge Y_2, & X_2 \wedge Y_3, & \neg X_2 \wedge Y_4, \\ \neg X_3 \wedge Y_1, & X_3 \wedge \neg Y_2, & X_3 \wedge \neg Y_3, & \neg X_3 \wedge \neg Y_4. \end{array}$$

The gadget reducing PC_1 to 2CSP can be obtained from this one by complementing all the occurrences of X_1.

Theorem 4.8 For every $\epsilon > 0$, MAX 2CSP is hard to approximate to within $9/10 + \epsilon$.

MAX 2CSP can be approximated within .859 [5]. The above theorem has implications for probabilistically checkable proofs. Reversing the well-known reduction from constraint satisfaction problems to probabilistically checkable proofs (cf. [1])[3], Theorem 4.8 yields the following theorem.

Theorem 4.9 For any $\epsilon > 0$, constants c and s exist such that $\mathsf{NP} \subseteq \mathsf{PCP}_{c,s}[\log, 2]$ and $c/s > 10/9 - \epsilon$.

The previously known gap between the completeness and soundness achievable reading two bits was 74/73 [2]. It would be $22/21 - \epsilon$ using Håstad's result [9] in combination with the argument of [2]. Actually the reduction from constraint satisfaction problems to probabilistically checkable proofs is reversible, and this will be important in Section 7.

RMBC **gadgets.** For any element of RMBC, there exists a 5-gadget reducing it to 2CSP, and it is optimal and strict. In this case we can found the gadget using our standard LP methodology, with no witness-restricting tricks, and thus with an implicit proof of optimality.

[3] The reverse connection is by now a folklore result and may be proved along the lines of [2, Proposition 10.3, Part (3)].

5 Interlude: Methodology

Despite their seeming variety, all the gadgets in this paper were computed using a single program (in the language APL2) to generate an LP, and call upon OSL (the IBM Optimization Subroutine Library) to solve it. This "gadget-generating" program takes several parameters.

The **source function** f is specified explicitly, by a small program that computes f.

The **target family** $\mathcal{F}$ is described by a single function, implemented as a small program, applied to all possible clauses of specified lengths and **symmetries**. The symmetries are chosen from among: whether clauses are unordered or ordered; whether their variables may be complemented; and whether they may include the constants 0 or 1. For example, a reduction to MAX CUT/0 would take as $\mathcal{F}$ the function $x_1 \oplus x_2$, applied over unordered binomial clauses, in which complementation is not allowed but the constant 0 is allowed. This means of describing $\mathcal{F}$ is relatively intuitive and has never restricted us, even though it is not completely general.

Finally, we specify an arbitrary set S of **selected assignments**, which allows us to search for S-partial gadgets (recall Definition 3.3). From equations (6) and (8), each selected assignment $\vec{a}$ generates a constraint that $(\exists \vec{b}) : \sum_j w_j C_j(\vec{a}, \vec{b}) = \alpha - (1 - f(\vec{a}))$. Selecting all *satisfying* assignments of f reproduces the set of constraints (2) for an α-gadget, while selecting *all* assignments reproduces the set of constraints (2) and (4) for a strict α-gadget.

Selected assignments are specified explicitly; by default, to produce an ordinary gadget, they are the satisfying assignments of f. The canonical witness for the selected set of assignments is generated by our program as governed by Definition 3.6. Notice that the definition of the witness depends on whether $\mathcal{F}$ is complementation-closed or not; and this is determined by the explicitly specified symmetries.

To facilitate the generation of **restricted witness matrices**, we have also made use of a "don't-care" state (in lieu of 0 or 1) to reduce the number of selected assignments. For example in reductions from RMBC_{00} we have used selected assignments of $(00*0)$ $(011*)$ $(10*0)$, and $(11*1)$. The various LP constraints must be satisfied for both values of any don't-care, while the witness function must not depend on the don't-care values. So in this example, use of a don't-care reduces the number of selected assignments from 8 to 4, reduces the number of auxiliary variables from about 2^8 to 2^4 (ignoring duplications of the 4 primary variables, or any symmetries), and reduces the number of constraints in the LP from 2^{2^8} (about 10^{77}) to 2^{2^4} (a more reasonable 65,536). Use of don't-cares provides a technique complementary to selecting a subset of all satisfying assignments, in that if the LP is feasible it provides an upper bound and a gadget, but the gadget may not be optimal.

In practice, selecting a subset of satisfying assignments has been by far the more useful of the two techniques; so far we have always been able to choose a subset which produces a lower bound and a gadget to match.

After constructing and solving an LP, the gadget-generating program uses brute force to make an independent verification of the gadget's validity, performance, and strictness.

The hardest computations were those for gadgets reducing from RMBC, which on an RS/6000 workstation took up to half an hour and memory of 500MB or so. However, the strength of [9] makes PC virtually the sole source function of contemporary interest, and all the reductions from PC can be run in seconds on a ThinkPad, with very modest memory requirements.

6 Improved Positive Results

In this section we show that we can use gadgets to improve approximation algorithms. In particular, we look at MAX 3SAT, and a variation, MAX 3ConjSAT, in which each clause is a conjunction (rather than a disjunction) of three literals. An improved approximation algorithm for the latter problem leads to improved results for probabilistically checkable proofs in which the verifier examines only 3 bits. Both of the improved approximation algorithms rely on strict gadgets reducing the problem to MAX 2SAT. We begin with some notation.

Definition 6.1 A (β_1, β_2)-approximation algorithm for MAX 2SAT is an algorithm which receives as input an instance with unary clauses of total weight m_1 and binary clauses of total weight m_2, and two reals $u_1 \leq m_1$ and $u_2 \leq m_2$, and produces reals $s_1 \leq u_1$ and $s_2 \leq u_2$ and an assignment satisfying clauses of total weight at least $\beta_1 s_1 + \beta_2 s_2$. If there exists an optimum solution that satisfies unary clauses of weight no more than u_1 and binary clauses of weight no more than u_2, then there is a guarantee that no assignment satisfies clauses of total weight more than $s_1 + s_2$.

That is, supplied with a pair of "upper bounds" u_1, u_2, a (β_1, β_2)-approximation algorithm produces a single upper bound of $s_1 + s_2$, along with an assignment respecting a lower bound of $\beta_1 s_1 + \beta_2 s_2$.

Lemma 6.2 [5] There exists a polynomial-time $(.976, .931)$-approximation algorithm for MAX 2SAT.

6.1 MAX 3SAT

In this section we show how to derive an improved approximation algorithm for MAX 3SAT. By restricting techniques in [8] from MAX SAT to MAX 3SAT and using a .931-approximation algorithm for MAX 2SAT due to Feige and Goemans [5], one can obtain a .7704-approximation algorithm for MAX 3SAT. The basic idea of [8] is to reduce each clause of length 3 to the three possible subclauses of length 2, give each new length-2 clause one-third the original weight, and then apply an approximation algorithm for MAX 2SAT. This approximation algorithm is then "balanced" with another approximation algorithm for MAX 3SAT to obtain the result. Here we show that by using a strict gadget to reduce 3SAT to MAX 2SAT, a good (β_1, β_2)-approximation algorithm for MAX 2SAT leads to a .801-approximation algorithm for MAX 3SAT.

Lemma 6.3 If for every $f \in$ E3SAT there exists a strict α-gadget reducing f to 2SAT, there exists a (β_1, β_2)-approximation algorithm for MAX 2SAT, and $\alpha \geq 1 + \frac{(\beta_1 - \beta_2)}{2(1-\beta_2)}$, then there exists a ρ-approximation algorithm for MAX 3SAT with

$$\rho = \frac{1}{2} + \frac{(\beta_1 - 1/2)(3/8)}{(\alpha - 1)(1 - \beta_2) + (\beta_1 - \beta_2) + (3/8)}.$$

Proof: Let ϕ be an instance of MAX 3SAT with length-1 clauses of total weight m_1, length-2 clauses of total weight m_2, and length-3 clauses of total weight m_3. We use the two algorithms listed below, getting the corresponding upper and lower bounds on number of satisfiable clauses:

- Random: We set each variable to 1 with probability 1/2. This gives a solution of weight at least $m_1/2 + 3m_2/4 + 7m_3/8$.
- Semidefinite programming: We use the strict α-gadget to reduce every length-3 clause to length-2 clauses. This gives an instance of MAX 2SAT. We apply the (β_1, β_2)-approximation

algorithm with parameters $u_1 = m_1$ and $u_2 = m_2 + \alpha m_3$ to find an approximate solution to this problem. The approximation algorithm gives an upper bound $s_1 + s_2$ on the weight of any solution to the MAX 2SAT instance and an assignment of weight $\beta_1 s_1 + \beta_2 s_2$. When translated back to the MAX 3SAT instance, the assignment has weight at least $\beta_1 s_1 + \beta_2 s_2 - (\alpha - 1)m_3$. Furthermore, $s_1 \leq m_1$, $s_2 \leq m_2 + \alpha m_3$, and the maximum weight satisfiable in the MAX 3SAT instance is at most $s_1 + s_2 - (\alpha - 1)m_3$.

The performance guarantee of the algorithm which takes the better of the two solutions is at least

$$\rho_1 \stackrel{\text{def}}{=} \min_{\substack{s_1 \leq m_1 \\ s_2 \leq m_2 + \alpha m_3}} \frac{\max\{m_1/2 + 3m_2/4 + 7m_3/8, \;\; \beta_1 s_1 + \beta_2 s_2 - (\alpha - 1)m_3\}}{s_1 + s_2 - (\alpha - 1)m_3}.$$

We now define a sequence of simplifications which will help prove the bound.

$$\begin{aligned}
\rho_2 &\stackrel{\text{def}}{=} \min_{\substack{t_1 \leq m_1 \\ t_2 \leq m_2 + m_3}} \frac{1}{t_1 + t_2} \max\{ && m_1/2 + 3m_2/4 + 7m_3/8, \\
& && \beta_1 t_1 + \beta_2 t_2 - (1 - \beta_2)(\alpha - 1)m_3\} \\
\rho_3 &\stackrel{\text{def}}{=} \min_{\substack{t_1 \leq m_1 \\ t_2 \leq m_2 + m_3}} \frac{1}{t_1 + t_2} \max\{ && t_1/2 + 3t_2/4 + m_3/8, \\
& && t_1/2 + 7m_3/8, \\
& && \beta_1 t_1 + \beta_2 t_2 - (1 - \beta_2)(\alpha - 1)m_3\} \\
\rho_4 &\stackrel{\text{def}}{=} \min_{t_2 \leq t} \frac{1}{t} \max\{ && t/2 + t_2/4 + m_3/8, \\
& && t/2 - t_2/2 + 7m_3/8, \\
& && \beta_1 t - (\beta_1 - \beta_2)t_2 - (1 - \beta_2)(\alpha - 1)m_3\} \\
\rho_5 &\stackrel{\text{def}}{=} \frac{1}{2} + \left(\frac{\frac{3}{8}(\beta_1 - \frac{1}{2})}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + \frac{3}{8}} \right) &&
\end{aligned}$$

To finish the proof of the lemma, we claim that

$$\rho_1 \geq \rho_2 \geq \cdots \geq \rho_5.$$

To see this, notice that the first inequality follows from the substitution of variables $t_1 = s_1$, $t_2 = s_2 - (\alpha - 1)m_3$. The second follows from the fact that setting m_1 to t_1 and m_2 to $\max\{0, t_2 - m_3\}$ only reduces the numerator. The third inequality follows from setting $t = t_1 + t_2$. The fourth is obtained by substituting a convex combination of the arguments instead of max and then simplifying. The convex combination takes a θ_1 fraction of the first argument, θ_2 of the second and θ_3 of the third, where

$$\theta_1 = \frac{\frac{2}{3}(1 - \beta_2)(\alpha - 1) + \frac{7}{6}(\beta_1 - \beta_2)}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + \frac{3}{8}},$$

$$\theta_2 = \frac{\frac{1}{3}(1 - \beta_2)(\alpha - 1) - \frac{1}{6}(\beta_1 - \beta_2)}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + \frac{3}{8}}$$

and

$$\theta_3 = \frac{\frac{3}{8}}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + \frac{3}{8}}.$$

Observe that $\theta_1 + \theta_2 + \theta_3 = 1$ and that the condition on α guarantees that $\theta_2 \geq 0$. $\square$

Remark 6.4 The analysis given in the proof of the above lemma is tight. In particular for an instance with m clauses such that

$$m_3 \stackrel{\text{def}}{=} m \frac{\beta_1 - 1/2}{(1-\beta_2)(\alpha-1) + (\beta_1 - \beta_2) + 3/8},$$

$m_1 \stackrel{\text{def}}{=} m - m_3$, $m_2 \stackrel{\text{def}}{=} 0$, $s_1 = m_1$, and $s_2 = \alpha m_3$, it is easy to see that $\rho_1 = \rho_5$.

The following lemma gives the strict gadget reducing functions in E3SAT to 2SAT. Notice that finding strict gadgets is almost as forbidding as finding gadgets for RMBC, since there are 8 existential constraints in the specification of a gadget. This time we relied instead on luck. We looked for an S-partial gadget for the set $S = \{111, 100, 010, 001\}$ and found an S-partial 3.5-gadget that turned out to be a gadget! Our choice of S was made judiciously, but we could have afforded to run through all $\binom{8}{4}$ sets S of size 4 in the hope that one would work.

Lemma 6.5 For every function $f \in$ E3SAT, there exists a strict (and optimal) 3.5-gadget reducing f to 2SAT.

Proof: Since 2SAT is complementation-closed, it is sufficient to present a 3.5-gadget reducing $(X_1 \vee X_2 \vee X_3)$ to 2SAT. The gadget is $X_1 \vee X_3, \neg X_1 \vee \neg X_3, X_1 \vee \neg Y, \neg X_1 \vee Y, X_3 \vee \neg Y, \neg X_3 \vee Y, X_2 \vee Y$, where every clause except the last has weight 1/2, and the last clause has weight 1. □

Combining Lemmas 6.2, 6.3 and 6.5 we get a .801-approximation algorithm.

Theorem 6.6 MAX 3SAT has a polynomial-time .801-approximation algorithm.

6.2 MAX 3-CONJ SAT

We now turn to the MAX 3ConjSAT problem. The analysis is similar to that of Lemma 6.3.

Lemma 6.7 If for every $f \in$ 3ConjSAT there exists a strict $(\alpha_1 + \alpha_2)$-gadget reducing f to 2SAT composed of α_1 length-1 clauses and α_2 length-2 clauses, and there exists a (β_1, β_2)-approximation algorithm for MAX 2SAT, then there exists a ρ-approximation algorithm for MAX 3ConjSAT with

$$\rho = \frac{\frac{1}{8}\beta_1}{\frac{1}{8} + (1-\alpha_1)(\beta_1 - \beta_2) + (1-\beta_2)(\alpha_1 + \alpha_2 - 1)}$$

provided $\alpha_1 + \alpha_2 > 1 + 1/8(1 - \beta_2)$.

Proof: Let ϕ be an instance of MAX 3ConjSAT with constraints of total weight m. As in the MAX 3SAT case, we use two algorithms and take the better of the two solutions:

- Random: We set every variable to 1 with probability half. The total weight of satisfied constraints is at least $m/8$.
- Semidefinite programming: We use the strict α-gadget to reduce any constraint to 2SAT clauses. This gives an instance of MAX 2SAT and we use the (β_1, β_2)-approximation algorithm with parameters $u_1 = \alpha_1 m$ and $u_2 = \alpha_2 m$. The algorithm returns an upper bound $s_1 + s_2$ on the total weight of satisfiable constraints in the MAX 2SAT instance, and an assignment of measure at least $\beta_1 s_1 + \beta_2 s_2$. When translated back to the MAX 3ConjSAT instance, the measure of the assignment is at least $\beta_1 s_1 + \beta_2 s_2 - (\alpha_1 + \alpha_2 - 1)m$. Furthermore, $s_1 \leq \alpha_1 m$, $s_2 \leq \alpha_2 m$, and the total weight of satisfiable constraints in the MAX 3ConjSAT instance is at most $s_1 + s_2 - (\alpha_1 + \alpha_2 - 1)m$.

Thus we get that the performance ratio of the algorithm which takes the better of the two solutions above is at least

$$\rho_1 \stackrel{\text{def}}{=} \min_{\substack{s_1 \le \alpha_1 m \\ s_2 \le \alpha_2 m}} \frac{\max\{m/8,\ \ \beta_1 s_1 + \beta_2 s_2 - (\alpha_1 + \alpha_2 - 1)m\}}{s_1 + s_2 - (\alpha_1 + \alpha_2 - 1)m}.$$

We now define a sequence of simplifications which will help prove the bound.

$$\rho_2 \stackrel{\text{def}}{=} \min_{\substack{t_1 \le \alpha_1 m \\ t_2 \le (1-\alpha_1)m}} \frac{1}{t_1 + t_2} \max\{m/8,\ \ \beta_1 t_1 + \beta_2 t_2 - (1-\beta_2)(\alpha_1 + \alpha_2 - 1)m\}$$

$$\rho_3 \stackrel{\text{def}}{=} \min_{\substack{t \le m \\ t_2 \le (1-\alpha_1)m}} \frac{1}{t} \max\{m/8,\ \ \beta_1 t - (\beta_1 - \beta_2)t_2 - (1-\beta_2)(\alpha_1 + \alpha_2 - 1)m\}$$

$$\rho_4 \stackrel{\text{def}}{=} \min_{t \le m} \frac{1}{t} \max\{m/8,\ \ \beta_1 t - ((1-\alpha_1)(\beta_1 - \beta_2) + (1-\beta_2)(\alpha_1 + \alpha_2 - 1))m\}$$

$$\rho_5 \stackrel{\text{def}}{=} \frac{\frac{1}{8}\beta_1}{\frac{1}{8} + (1-\alpha_1)(\beta_1 - \beta_2) + (1-\beta_2)(\alpha_1 + \alpha_2 - 1)}$$

In order to prove the lemma, we claim that

$$\rho_1 \ge \rho_2 \ge \cdots \ge \rho_5.$$

To see this, observe that the first inequality follows from the substitution of variables $t_1 = s_1$ and $t_2 = s_2 - (\alpha_1 + \alpha_2 - 1)m$. The second follows from setting $t = t_1 + t_2$. The third inequality follows from the fact that setting t_2 to $(1 - \alpha_1)m$ only reduces the numerator. The fourth is obtained by substituting a convex combination of the arguments instead of max and then simplifying. $\square$

The following gadget was found by looking for an S-partial gadget for $S = \{111, 110, 101, 011\}$.

Lemma 6.8 For any $f \in$ 3ConjSAT there exists a strict (and optimal) 4-gadget reducing f to 2SAT. The gadget is composed of one length-1 clause and three length-2 clauses.

Proof: Recall that 2SAT is complementation-closed, and thus it is sufficient to exhibit a gadget reducing $f(a_1, a_2, a_3) = a_1 \wedge a_2 \wedge a_3$ to 2SAT. Such gadget is Y, $(\neg Y \vee X_1)$, $(\neg Y \vee X_2)$, $(\neg Y \vee X_3)$, where all clauses have weight 1. The variables X_1, X_2, X_3 are primary variables and Y is an auxiliary variable. $\square$

Theorem 6.9 MAX 3ConjSAT has a polynomial-time .367-approximation algorithm.

It is shown by Trevisan [16, Theorem 18] that the above theorem has consequences for $\mathsf{PCP}_{c,s}[\log, 3]$. This is because the computation of the verifier in such a proof system can be described by a decision tree of depth 3, for every choice of random string. Further, there is a 1-gadget reducing every function which can be computed by a decision tree of depth k to kConjSAT.

Corollary 6.10 $\mathsf{PCP}_{c,s}[\log, 3] \subseteq \mathsf{P}$ provided that $c/s > 2.7214$.

The previous best trade-off between completeness and soundness for polynomial-time PCP classes was $c/s > 4$ [16].

7 Lower Bounds for Gadget Constructions

In this section we shall show that some of the gadget constructions mentioned in this paper and in [2] are optimal, and we shall prove lower bounds for some other gadget constructions.

The following result is useful to prove lower bounds for the RMBC family.

Lemma 7.1 If there exists an α-gadget reducing an element of RMBC to a complementation-closed constraint family $\mathcal{F}$, then there exists an α-gadget reducing all elements of PC to $\mathcal{F}$.

Proof: If a family $\mathcal{F}$ is complementation-closed, then an α-gadget reducing an element of PC (respectively RMBC) to $\mathcal{F}$ can be modified (using complementations) to yield α-gadgets reducing all elements of PC (respectively RMBC) to $\mathcal{F}$. For this reason, we will restrict our analysis to PC_0 and RMBC_{00} gadgets. Note that, for any $a_1, a_2, a_3 \in \{0,1\}^3$, $\mathrm{PC}_0(a_1, a_2, a_3) = 1$ if and only if $\mathrm{RMBC}_{00}(a_1, a_2, a_3, \overline{a_3}) = 1$. Let Γ be an α gadget over primary variables $x_1, \ldots, x_4$ and auxiliary variables $y_1, \ldots, y_n$ reducing RMBC to 2SAT. Let Γ' be the gadget obtained from Γ by imposing $x_4 \equiv \overline{x_3}$: it is immediate to verify that Γ' is an α-gadget reducing PC_0 to $\mathcal{F}$. □

7.1 Reducing PC and RMBC to 2SAT

Theorem 7.2 If Γ is an α-gadget reducing an element of PC to 2SAT, then $\alpha \geq 11$.

Proof: It suffices to consider PC_0. We prove that the optimum of (LP1) is at least 11. To this end, consider the dual program of (LP1). We have a variable $y_{\vec{a},\vec{b}}$ for any $\vec{a} \in \{0,1\}^3$ and any $\vec{b} \in \{0,1\}^4$, plus additional variables $\hat{y}_{\vec{a},\vec{b}^{opt}(\vec{a})}$ for any $\vec{a} : \mathrm{PC}(\vec{a}) = 1$, where $\vec{b}^{opt}$ is the "optimal" witness function defined in Section 3. The formulation is

$$
\begin{array}{lll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} & \\
\text{subject to} & & \\
& \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} y_{\vec{a},\vec{b}^{opt}(\vec{a})} & \\
& \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} \hat{y}_{\vec{a},\vec{b}^{opt}(\vec{a})} C_j(\vec{a},\vec{b}^{opt}(\vec{a})) & \forall j \in [98] \\
& y_{\vec{a},\vec{b}} \geq 0 & (\forall \vec{a} \in \{0,1\}^3)\ (\forall \vec{b} \in \{0,1\}^4) \\
& \hat{y}_{\vec{a},\vec{b}^{opt}(\vec{a})} \geq 0 & (\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)
\end{array}
\tag{DUAL1}
$$

There exists a feasible solution for (DUAL1) whose cost is 11. □

Corollary 7.3 If Γ is an α-gadget reducing an element of RMBC to 2SAT, then $\alpha \geq 11$.

7.2 Reducing PC and RMBC to SAT

Theorem 7.4 If Γ is an α-gadget reducing an element of PC to SAT, then $\alpha \geq 4$.

Proof: As in the proof of Theorem 7.2 we give a feasible solution to the dual to obtain the lower bound. The linear program that finds the best gadget reducing PC_0 to SAT is similar to (LP1), the only difference being that a larger number N of clauses are considered, namely, $N = \sum_{i=1}^{7} \binom{7}{i} 2^i$.

The dual program is then

$$
\begin{array}{lll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} & \\
\text{subject to} & & \\
& \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} y_{\vec{a},\vec{b}^{opt}(\vec{a})} & \\
& \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} \hat{y}_{\vec{a},\vec{b}^{opt}(\vec{a})} C_j(\vec{a},\vec{b}^{opt}(\vec{a})) & \forall j \in [N] \\
& y_{\vec{a},\vec{b}} \geq 0 & (\forall \vec{a} \in \{0,1\}^3)\ (\forall \vec{b} \in \{0,1\}^4) \\
& \hat{y}_{\vec{a},\vec{b}^{opt}(\vec{a})} \geq 0 & (\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)
\end{array}
\tag{DUAL2}
$$

Consider now the following assignment of values to the variables of (DUAL2) (the unspecified values have to be set to zero):

$$
\begin{array}{l}
(\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)\ \hat{y}_{\vec{a},\vec{b}^{opt}(\vec{a})} = \frac{3}{4} \\
(\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)(\forall \vec{a}' : d(\vec{a},\vec{a}') = 1)\ y_{\vec{a}',\vec{b}^{opt}(\vec{a})} = \frac{1}{3}
\end{array}
$$

where d is the Hamming distance between binary sequences. It is possible to show that this is a feasible solution for (DUAL2) and it is immediate to verify that its cost is 4. $\square$

Corollary 7.5 If Γ is an α-gadget reducing an element RMBC to SAT, then $\alpha \geq 4$.

7.3 Reducing PC and RMBC to CUT and DICUT

The optimality of the gadgets reducing PC to CUT/0 and DICUT is given below.

Theorem 7.6 If Γ is an α-gadget reducing PC_0 to CUT/0, then $\alpha \geq 8$. If Γ is an α-gadget reducing PC_1 to CUT/0, then $\alpha \geq 9$.

Theorem 7.7 If Γ is an α-gadget reducing an element of PC to DICUT, then $\alpha \geq 6.5$.

The optimality of gadgets reducing RMBC to CUT and DICUT were established using the idea of S-partial gadgets. For example the optimality of the 8-gadget reducing RMBC_{00} to CUT was established using $S = \{0000, 0111, 1101, 1010\}$ where the 4-tuples give the assignment to the variables a, b, c, d (in that order) when applied to the function $RMBC_{00}(a, b, c, d)$.

7.4 Reducing kSAT to lSAT

Let k and l be any integers $k > l \geq 3$. The standard reduction from EkSAT to lSAT can be seen as a $\lceil (k-2)/(l-2) \rceil$-gadget. In this section we shall show that this is asymptotically the best possible. Note that since lSAT is complementation-closed we can restrict ourselves to considering just one constraint function of EkSAT, say $f(a_1, \ldots, a_k) \equiv \bigvee_i a_i$.

Theorem 7.8 For any $k > l > 2$, if Γ is an α-gadget reducing f to lSAT then $\alpha \geq k/l$.

Proof: We can write a linear program whose optimum gives the smallest α such that an α-gadget exists reducing f to lSAT. Let b be the witness function used to formulate this linear program. We can assume that b is 2^{2^k}-ary and we let $K = 2^{2^k}$. Also let N be the total number of constraints from lSAT that can be defined over $k + K$ variables. Assume some enumeration $C_1, \ldots, C_N$ of such constraints. The dual LP is

$$
\begin{array}{ll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} \\
\text{subject to} & \\
 & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:f(\vec{a})=1} y_{\vec{a},\vec{b}^{kSAT-lSAT}(\vec{a})} \\
\forall j \in [N] & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:f(\vec{a})=1} \hat{y}_{\vec{a},\vec{b}^{kSAT-lSAT}(\vec{a})} C_j(\vec{a},\vec{b}^{kSAT-lSAT}(\vec{a})) \\
\forall \vec{a} \in \{0,1\}^k,\ \forall \vec{b} \in \{0,1\}^K & y_{\vec{a},\vec{b}} \geq 0 \\
\forall \vec{a} : f(\vec{a}) = 1 & \hat{y}_{\vec{a},\vec{b}^{kSAT-lSAT}(\vec{a})} \geq 0
\end{array}
\tag{DUAL3}
$$

The witness function $\vec{b}^{kSAT-lSAT}$ is an "optimal" witness function for gadgets reducing kSAT to lSAT.

Let $A_k \subset \{0,1\}^k$ be the set of binary k-ary strings with exactly one non-zero component (note that $|A_k| = k$). Also let $\vec{0}$ (respectively, $\vec{1}$) be the k-ary string all whose components are equal to 0 (respectively, 1). The following is a feasible solution for (DUAL3) whose cost is k/l. We only specify non-zero values.

$$
\begin{array}{ll}
(\forall \vec{a} \in A_k) & \hat{y}_{\vec{a},\vec{b}^{kSAT-lSAT}(\vec{a})} = 1/l \\
(\forall \vec{a} \in A_k) & y_{\vec{0},\vec{b}^{kSAT-lSAT}(\vec{a})} = 1/l \\
(\forall \vec{a} \in A_k) & y_{\vec{1},\vec{b}^{kSAT-lSAT}(\vec{a})} = 1/k \ .
\end{array}
$$

$\square$

In view of the above lower bound, a gadget cannot provide an approximation-preserving reduction from MAX SAT to MAX kSAT. More generally, there cannot be an approximation-preserving gadget reduction from MAX SAT to, say, MAX $(\log n)$SAT. In partial contrast with this lower bound, Khanna et al. [13] have given an approximation-preserving reduction from MAX SAT to MAX 3SAT and Crescenzi and Trevisan [4] have provided a *tight* reduction between MAX SAT and MAX $(\log n)$SAT, showing that the two problems have the same approximation threshold.

Acknowledgements

We thank Pierluigi Crescenzi and Oded Goldreich for several helpful suggestions and remarks. We are grateful to John Forrest and David Jensen for their assistance in efficiently solving large linear programs. We thank Howard Karloff and Uri Zwick for pointing out the error in the earlier version of this paper, and the counterexample to our earlier claim. We thank the anonymous referees for their numerous comments and suggestions leading to the restructuring of Section 3.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM* 45(3):501–555, 1998.

[2] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs and nonapproximability – towards tight results. *SIAM J. on Computing* 27(3):804–915, 1998.

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

[4] P. Crescenzi and L. Trevisan. MAX NP-completeness made easy. Manuscript, 1996.

[5] U. Feige and M. X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proc. of the 3rd Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.

[6] M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[7] M. X. Goemans and D. P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994.

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

[9] J. Håstad. Some optimal inapproximability results. In *Proc. of the 29th ACM Symposium on Theory of Computing*, pages 1–10, 1997.

[10] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proc. of the 38th IEEE Symposium on Foundations of Computer Science*, pages 406–415, 1997.

[11] H. Karloff and U. Zwick. Personal communication, 1996.

[12] R. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, Advances in Computing Research, pages 85–103. Plenum Press, 1972.

[13] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM J. on Computing* 28(1):164–191, 1999.

[14] T. Ono, T. Hirata, and T. Asano. Approximation algorithms for the maximum satisfiability problem. In *Proc. of the 5th Scandinavian Workshop on Algorithm Theory*, pages 100–111. LNCS 1097, Springer-Verlag, 1996.

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

[16] L. Trevisan. Parallel approximation algorithms using positive linear programming. *Algorithmica*, 21:72–88, 1998.

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

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

[19] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proc. of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 201–210, 1998.