# Gadgets, Approximation, and Linear Programming

[Extended Abstract]

Luca Trevisan [*] Gregory B. Sorkin [†]

Madhu Sudan[†] David P. Williamson[†]

## Abstract

We present a linear-programming based method for finding "gadgets", i.e., combinatorial structures reducing constraints of one optimization problems 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 [1] on how to prove the optimality of gadgets — we show how LP duality gives such proofs.

The new gadgets improve hardness results for MAX CUT and MAX DICUT, showing that approximating these problems to within factors of 60/61 and 44/45 respectively is NP-hard (improving upon the previous hardness of 71/72 for both problems [1]). 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 in [6, 3]) 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 typical example is in the reduction from 3SAT to MAX 2SAT [4] in which a clause $C_k = X_1 \vee X_2 \vee X_3$ is replaced by ten clauses

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

If clause $C_k$ is satisfied, then 7 of the 10 new clauses are satisfied by setting $Y^k$ appropriately; otherwise only 6 of the 10 are satisfiable. Use of the superscript $Y^k$ is only to indicate that each clause $C_k$ gets its own "auxiliary variables"; henceforth we will only consider one clause (or one "constraint") at a time, so we'll dispense with the superscripts. We will re-visit the 3SAT-to-2SAT reduction in Lemma 6.4.

Starting with the work of Karp, 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 non-approximability of optimization problems.

Despite their importance, the construction of gadgets has always been a "black art", with no known uniform methods. In fact, until recently no one had even proposed a concrete definition of a gadget; Bellare, Goldreich and Sudan [1] 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, finding a gadget for a given reduction remained an ad hoc task. Additionally, it remained hard to prove that a gadget's cost was optimal.

This paper addresses the two issues raised above. 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 impractically slow, and search-based proofs of the optimality (or the non-existence) of a gadget would be monstrously large.

As the next step, we show how to express the search

[*] Dipartimento di Scienze dell'Informazione, Università degli Studi di Roma "La Sapienza", Via Salaria 113, 00198 Rome, Italy. trevisan@dsi.uniroma1.it. Partially supported by a project of the Italian Ministero della Università e della Ricerca Sientifica e Tecnologica.

[†] IBM T.J. Watson Research Center, Yorktown Heights NY 10598. {sorkin, madhu, dpw}@watson.ibm.com.

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. (The latter would yield an integer program (IP) while the former yields an LP). This seemingly helps only in showing hardness of weighted optimization problems, but a new result due to Crescenzi, Silvestri and Trevisan [2] 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 the weighted version is as good as working with the 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. [1] presented gadgets reducing the computation of a verifier to several problems, including MAX 3SAT, MAX 2SAT, and MAX CUT. We examine these in turn and show that the gadgets in [1] for MAX 3SAT and MAX 2SAT were optimal, but their MAX CUT gadget was 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 its hardness. Our final result, obtained by plugging our gadget into the proof of [1], 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. (For both problems, the previous best hardness factor was 71/72 [1].)[2]

Obtaining better reductions between problems can also yield improved approximation algorithms for some problems (if the reduction goes the right way!). We illustrate the point by constructing a gadget reducing MAX 3SAT to MAX 2SAT. Using this new reduction in combination with a technique of Goemans and Williamson [5, 6] and the state-of-the-art .931-approximation algorithm for MAX 2SAT due to Feige and Goemans [3] (which improves upon the previous (famous) .878-approximation algorithm of [6]), we obtain a .801-approximation algorithm for MAX 3SAT. The best result that could be obtained previously, by combining the technique of [5, 6] and the bound of [3], was .7704. (This is not mentioned explicitly anywhere but why would we lie. See also the .769-approximation algorithm in the paper of Ono, Hirata, and Asano [8].)

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, that there exist constants $c$ and $s$, $c/s > 34/33$, 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 previously known bounds for these results were 74/73 [1] and 4 [9] respectively.

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), but it still relies on defining an LP which describes the optimal gadget, and extracting a lower bound from the LP's dual.

**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 an example of a proof of the optimality of a gadget.

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

[2]Note that 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 is the reciprocal of the factors mentioned in [1] and exactly the factors as stated in [10, 5, 6, 3].

## 2 Definitions

We begin with some definitions we will need before giving the definition of a gadget from [1].

**Definition 2.1** 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 $f(X_1, \ldots, X_k) = 1$. In the opening example, the reduction from 3SAT to MAX 2SAT, $f$ is the constraint $X_1 \vee X_2 \vee X_3$.

**Definition 2.2** 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 $\mathcal{F}$.

In the reduction from 3SAT to MAX 2SAT, $\mathcal{F}$ contains the three binary (2-ary) constraint functions $f_{00}(a_1, a_2) = a_1 \vee a_2$, $f_{10}(a_1, a_2) = \neg a_1 \vee a_2$, and $f_{11}(a_1, a_2) = \neg a_1 \vee \neg a_2$, and the two unary (1-ary) constraint functions $f_0(a_1) = a_1$ and $f_1(a_1) = \neg a_1$. How these are applied is formalized in the next definition.

**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 $i_j \in [n]$ for $j \in [k]$. Variable $X_j$ is said to *occur* in $C$ if $j \in \{i_1, \ldots, i_k\}$. 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 $f \in F$.

So in the opening example, the first constraint $\neg X_1$ could be described as $(f_1, (1))$, while (if we give $Y$ the index 4), the last clause $\neg X_3 \vee \neg Y$ is the constraint $(f_{11}, (3, 4))$.

We can now formally define a gadget.

**Definition 2.4** [Gadget [1]] For $\alpha \in \mathcal{R}^+$, a constraint function $f : \{0,1\}^k \to \{0,1\}$, and a constraint family $\mathcal{F}$: an $\alpha$*-gadget* (or "gadget with performance $\alpha$") reducing $f$ to $\mathcal{F}$ is 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:

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

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

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

The gadget is *strict* if, in addition,

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

We say that the function $\vec{b} = \vec{b}(\vec{a})$ is a *witness* for the gadget if equation (2) (and, for a strict gadget, equation (4)) is satisfied by $\vec{b}(\vec{a})$. For a given witness function $\vec{b}$, the function $b_i(\vec{a}) = \vec{b}(\vec{a})_i$.

To show that the introductory example is a strict 7-gadget, one witness (it is not unique) is the function $\vec{b}(a_1, a_2, a_3) = a_1 \wedge a_2 \wedge a_3$. It is straightforward to verify that this satisfies 7 clauses if one, two, or all three of $a_1, a_2$, and $a_3$ are 1, and satisfies 6 clauses if $a_1 = a_2 = a_3 = 0$; it is a separate step to verify that no satisfying assignment gives a value exceeding 7, and no unsatisfying assignment gives a value exceeding 6.

We observe that because the weights may all be re-scaled, what is significant is the ratio of the right-hand sides of equations (1–4). A "strong" gadget is one with a small $\alpha$; in particular, if $\alpha' > \alpha$, any $\alpha$-gadget is also automatically (after re-scaling) an $\alpha'$-gadget. (But strictness is not maintained.) In the extreme, a 1-gadget would do a perfect reduction.

Because of the natural association between a gadget and the corresponding function $\sum w_j C_j$, a gadget may be thought of as an instance of a "(weighted) constraint satisfaction problem".

**Definition 2.5** For a function family $\mathcal{F}$, *MAX* $\mathcal{F}$ is the optimization problem whose instances consist of $m$ constraints from $\mathcal{F}$, on $n$ variables, and whose objective is to find an assignment to the variables which maximizes the number of satisfied constraints.

Viewed as a constraint satisfaction problem, a gadget has the property that when restricted to any satisfying assignment $\vec{a}$ to $X_1, \ldots, X_k$ its maximum is exactly $\alpha$, and when restricted to any unsatisfying assignment its maximum is at most $\alpha - 1$ (exactly $\alpha - 1$ if the gadget is strict).

For convenience we now give a comprehensive list of all the constraints and constraint families used in this paper. We motivate these classes briefly after the definitions.

**Definition 2.6**

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

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

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

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

  $\mathrm{RMBC}_{00}$ may be thought of as the test $(c,d)[a] \overset{?}{=} b$, $\mathrm{RMBC}_{01}$ as the test $(c,\neg d)[a] \overset{?}{=} b$, $\mathrm{RMBC}_{10}$ as the test $(\neg c, d)[a] \overset{?}{=} b$ and $\mathrm{RMBC}_{11}$ as the test $(c,d)[a] \overset{?}{=} \neg b$.

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

- For any $k \geq 1$, *k-SAT* is the constraint family $k\mathrm{SAT} = \bigcup_{l \in [k]} \mathrm{E}l\mathrm{SAT}$.

- *SAT* is the constraint family $\mathrm{SAT} = \bigcup_{l \geq 1} \mathrm{E}l\mathrm{SAT}$

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

- *CUT* is the constraint function $\mathrm{CUT} : \{0,1\}^2 \to \{0,1\}$ with $\mathrm{CUT}(a,b) = a \oplus b$.

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

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

Table 1 summarizes the gadgets found in this paper. The gadget reduces a function $f$ (called source) to a family $\mathcal{F}$ (called target). The above list of constraint families includes both sources and targets of reductions.

Our interest in the function families PC and RMBC comes from the following theorem of [1].

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

Thus using CUT, DICUT, 2CSP, E$k$SAT and $k$SAT as the target of reductions shows the hardness of MAX CUT, MAX DICUT, MAX 2CSP, MAX E$k$SAT and MAX $k$SAT respectively, and minimizing the value of $\alpha$ in the gadgets gives better hardness results.

We also use 2SAT as a target to obtain new approximation algorithms for the source (by a reduction to MAX 2SAT and using the algorithm of [3] to approximate this problem). The two families reduced to 2SAT in this way are 3SAT and 3ConjSAT.

## 3 The Basic Procedure

In this section we shall illustrate our technique by constructing a gadget reducing $\mathrm{PC}_0$ to 2SAT.

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*.

Let $f$ be a $k$-ary function with $s$ satisfying assignments $\vec{a}^{(1)}, \ldots, \vec{a}^{(s)}$. For a gadget $\Gamma$ with $n$ auxiliary variables reducing $f$ to a family $\mathcal{F}$, an *extended witness* of $\Gamma$ is a $s \times (n+k)$ matrix $(c_{ij})$ such that $c_{ij} = a_j^{(i)}$ for $j \leq k$, and $c_{ij} = b_{j-k}(\vec{a}^{(i)})$ otherwise, where $\vec{b}$ is a witness of $\Gamma$.

**Lemma 3.1** If an $\alpha$-gadget $\Gamma$ reducing $f$ to an $r$-ary family $\mathcal{F}$ has an extended witness with $r+1$ equal columns, and at least one column corresponds to an auxiliary variable, then there is an $\alpha$-gadget $\Gamma'$ using one fewer auxiliary variable. If $\Gamma$ is strict, so is $\Gamma'$.

*Proof*: Let $(c_{ij})$ be the extended witness claimed in the hypothesis, let $i_1, \ldots, i_r, i_{r+1}$ be indices of equal columns of $(c_{ij})$, and assume without loss of generality that $i_{r+1} > k$. We wish to show that we can obtain a new $\alpha$-gadget by replacing occurrences of $X_{i_{r+1}}$ with some other $X_{i_j}$. For any constraint $C$ of $\Gamma$, define $red(C)$ as follows. If $X_{i_{r+1}}$ does not occur in $C$, then $red(C) = C$. Otherwise, since at most $r$ variables occur in $C$, it follows that $X_{i_h}$ does not occur in $C$ for some $h \in [k]$. Then, we define $red(C)$ as the constraint obtained from $C$ by replacing the occurrence of $X_{i_{r+1}}$ by an occurrence of $X_{i_h}$. If $\Gamma = (C_1, \ldots, C_m, w_1, \ldots, w_m)$, then define a new gadget $\Gamma' = (red(C_1), \ldots, red(C_m), w_1, \ldots, w_m)$. Correspondingly, let $\vec{b}'(\vec{a})$ be identical to $b(\vec{a})$ but with $b_{i_{r+1}}$ eliminated. $\Gamma'$ has $n-1$ auxiliary variables ($X_{i_{r+1}}$ never occurs in $\Gamma'$). By construction, $\Gamma'(\vec{a}, \vec{b}'(\vec{a})) \equiv \Gamma(\vec{a}, \vec{b}(\vec{a}))$, so $\Gamma'$ satisfies the gadget-defining equation (2). (Similarly, for strict gadgets $\Gamma$,

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

Table 1: All gadgets described are provably optimal, and strict. The sole exception (†) is the best possible strict gadget; there is a non-strict 3-gadget. All previous results quoted are interpretations of the results in [1], except the gadget reducing 3SAT to 2SAT, which is due to [4], and the gadget reducing PC to 3SAT, which is folklore. The gadgets marked with (‡) are not strictly reductions to CUT; see Section 4.1.

$\Gamma'$ satisfies (4)). Also, the range of the universal quantification for $\Gamma'$ is smaller than that for $\Gamma$, therefore $\Gamma'$ satisfies inequalities (1) and (3). □

**Definition 3.2** For a constraint $f$, call two variables $a_{j'}$ and $a_j$ *distinct* if there exists an assignment $\vec{a}$, satisfying $f$, for which $a_{j'} \neq a_j$.

**Corollary 3.3** Suppose $f$ is a constraint on $k$ variables, with $s$ satisfying assignments and $k'$ distinct variables. If there is an $\alpha$-gadget reducing $f$ to an $r$-ary family $\mathcal{F}$, then there is an $\alpha$-gadget with at most $r2^s - k'$ auxiliary variables.

If there is a strict $\alpha$-gadget reducing $f$ to $\mathcal{F}$, then there is a strict $\alpha$-gadget with at most $r2^{2^k} - k$ auxiliary variables.

*Proof*: In the first case, the domain of the witness function $b$ is $\{a : f(a) = 1\}$, a set of cardinality $s$, so an extended witness has $s$ rows, and the number of distinct columns is at most $2^s$. If there are more than $r+1$ equal columns, and not all of them correspond to primary variables, then we can eliminate one auxiliary variable as per the preceding lemma; thus no more than $r2^s - k'$ auxiliary variables are required.

For strict gadgets the proof is identical, except that the domain of the witness function $b$ is $\{a \in \{0,1\}^k\}$. Here the $k$ primary variables are all distinct, since the domain considered is that of all assignments. □

Note that further reductions in the number of auxiliary variables are often possible. In the proof of Lemma 3.1 we used substitutions whenever there were $r+1$ equal columns. Indeed, for constraint families like 3SAT we can make substitutions whenever there are *two* equal columns, since there is the possibility to replace an (illegal) constraint like $(X_1 \vee X_1 \vee X_3)$ by a legal constraint $(X_1 \vee X_3)$. In general this is possible if the target of the reduction is a family with a property that we name by analogy with the terminology for matroids:

**Definition 3.4** A constraint family $\mathcal{F}$ is *hereditary* if for any $f_i(X_1, \ldots, X_{n_i}) \in \mathcal{F}$, and any two indices $j, j' \in [n_i]$, the function $f_i$ when restricted to $X_j \equiv X_{j'}$ and considered as a function of $n_i - 1$ variables, is identical (up to the order of the arguments) to some other function $f_{i'} \in \mathcal{F} \cup \{0,1\}$, where $n_{i'} = n_i - 1$ (and 0 and 1 denote the constant functions).

**Lemma 3.5** If an $\alpha$-gadget $\Gamma$ reducing $f$ to a hereditary family $\mathcal{F}$ has a witness function for which two auxiliary variables are identical (i.e. $b_{j'}(\cdot) \equiv b_j(\cdot)$), or if an auxiliary variable is identical to a primary variable ($b_{j'}(\vec{a}) \equiv a_j$) then there is an $\alpha$-gadget $\Gamma'$ using one fewer auxiliary variable. If $\Gamma$ is strict, so is $\Gamma'$.

*Proof*: Similar to the proof of Lemma 3.1, except we use the hereditary property to ensure that the result of substitution is a gadget. □

If the constraint family allows the complementation of any variable (as for example 2SAT but not CUT or DICUT), then the number of auxiliary variables may be approximately halved: we need only consider witness functions whose value is 1 at least as often as it is 0.

**Definition 3.6** A family $\mathcal{F}$ is *complementation-closed* if it is hereditary and, for any $f_i(X_1, \ldots, X_{n_i}) \in \mathcal{F}$, and any index $j \in [n_i]$, the function $f'_i$ given by $f'_i(X_1, \ldots, X_{n_i}) = f_i(X_1, \ldots, X_{j-1}, \neg X_j, X_{j+1}, \ldots, X_{n_i})$ is contained in $\mathcal{F}$.

Notice that for a complementation-closed family $\mathcal{F}$, the hereditary property implies that if $f_i(X_1, \ldots, X_{n_i})$ is contained in $\mathcal{F}$ then so is the function $f_i$ restricted to $X_j \equiv \neg X_{j'}$ for any two distinct indices $j, j' \in [n_i]$. This guarantees that we can make substitutions even if two columns in the (extended) witness are complements of each other. To sum up, we have the following result.

**Lemma 3.7** Suppose $f$ is a constraint on $k$ variables, with $s$ satisfying assignments and $k' \leq k$ variables distinct even under complementation. If there is an $\alpha$-gadget reducing $f$ to a hereditary (respectively, complementation-closed) constraint family $\mathcal{F}$, then there is an $\alpha$-gadget with at most $2^s - k'$ (respectively, $2^{s-1} - k'$) auxiliary variables.

In some cases (e.g. 2SAT but not CUT), there is also no need to consider witness functions which are identically 0 or identically 1. (Clauses in which their corresponding auxiliary variables appear can be replaced by shorter clauses eliminating those variables.)

The previous discussion means that if we are looking for an $\alpha$-gadget reducing $\mathrm{PC}_0$ to 2SAT with the minimum value of $\alpha$, then we can restrict our search to gadgets over at most 7 variables. Over 7 variables, $2 \cdot 7 + 4 \cdot \binom{7}{2}$ 2SAT constraints can be defined; call them $C_1, \ldots, C_{98}$. A gadget over 7 variables can thus be identified with the vector $(w_1, \ldots, w_{98})$ of the weights of the constraints. Since in [1] it is shown that an 11-gadget exists reducing $\mathrm{PC}_0$ to 2SAT, it follows that in an optimum gadget no constraint will have a weight larger than 11. If we were allowed only integer weights over the constraints, then an optimum gadget could be found by exhaustive search over a space of $12^{98}$ elements. Allowing real weights over the constraints makes the search space infinite, yet we can use linear programming to find an optimum gadget quite efficiently. Consider the following linear program with $64 + 4 + 64 = 132$ constraints over 99 variables:

$$\begin{array}{lrcl}
\text{minimize } \alpha & & & \qquad\qquad (\text{LP1}) \\
\text{subject to} & & & \\
(\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 1)\ (\forall \vec{b}) : & \sum_j w_j C_j(\vec{a}, \vec{b}) & \leq & \alpha \\
(\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 1) : & \sum_j w_j C_j(\vec{a}, \vec{b}(\vec{a})) & = & \alpha \\
(\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 0)\ (\forall \vec{b}) : & \sum_j w_j C_j(\vec{a}, \vec{b}) & \leq & \alpha - 1 \\
 & \alpha & \geq & 0 \\
(\forall j \in [98]) : & w_j & \geq & 0.
\end{array}$$

In general, any "duplicated" variables can be eliminated from an $\alpha$-gadget to give a "simplified" $\alpha'$-gadget ($\alpha' \leq \alpha$), and the $\alpha'$-gadget can be associated with its (fixed-length) vector of weights.

**Theorem 3.8** The weight vector associated with any simplified $\alpha$-gadget provides a feasible solution to the associated LP, and conversely any feasible solution to the LP is a weight vector which describes an $\alpha$-gadget. An optimal LP solution yields an optimal $\alpha$-gadget (one where $\alpha$ is as small as possible).

In particular, (LP1) has optimal solution $\alpha = 11$, proving the optimality of the [1] gadget.

In the remaining sections we give applications of some gadgets and then report their best possible gadgets. All gadgets are computer-constructed unless otherwise noted. Most gadgets are omitted for brevity.

## 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 2.4 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 the gadget's value is still $\alpha$, violating (3)). Following [1], we generalize the definition:

**Definition 4.1** A *gadget with auxiliary constant 0* is a gadget as previously defined, except that (1–4) are only required to hold when $Y_1 = 0$.

To get a hardness result for MAX CUT, we first need the following lemma, which is a very minor modification of a lemma in [1].

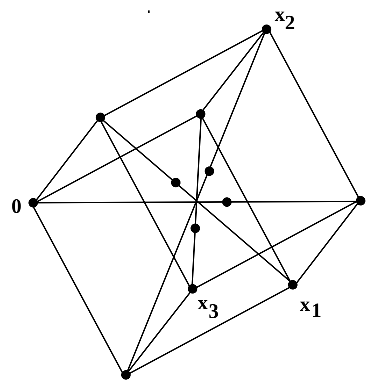

Figure 1: 8-gadget reducing $PC_0$ to CUT. Every edge has weight .5. The auxiliary variable which is always 0 is labelled 0.

**Lemma 4.2** [1] For the constraint family CUT, if there exists an $\alpha_1$-gadget with constant 0 reducing every function in PC to CUT and an $\alpha_2$-gadget with constant 0 reducing every function in RMBC to CUT, then for any $\gamma > 0$, MAX CUT is NP-hard to approximate to within $1 - \frac{.15}{.6\alpha_1 + .4\alpha_2} + \gamma$.

Notice that the CUT constraint family is hereditary, since identifying the two variables in a CUT constraint yields the constant function 0. Thus by Lemma 3.7, if there is an $\alpha$-gadget with constant 0 reducing $PC_0$ to CUT, then there is an $\alpha$-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 $\alpha$-gadget with constant 0 reducing $PC_0$ to CUT. A linear program of the same size can similarly be constructed to find a gadget with constant 0 reducing $PC_1$ to CUT.

**Lemma 4.3** There exists an 8-gadget with constant 0 reducing $PC_0$ to CUT, and it is optimal and strict. There exists a 9-gadget reducing $PC_1$ to CUT, and it is optimal and strict.

The $PC_0$ gadget is shown in Figure 1. It turns out that we cannot apply exactly the technique above to find an optimal gadget that reduces $RMBC_{00}$ to CUT. Since there are 8 satisfying assignments to the 4 variables of the $RMBC_{00}$ constraint, by Lemma 3.7, 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 computers.

To overcome this difficulty, we observe that for the $RMBC_{00}$ function when $a_1 = 0$, the value of $a_4$ is irrelevant, and when $a_1 = 1$, the value of $a_3$ is irrelevant. Thus we consider only restricted witness functions for which $\vec{b}(0, a_2, a_3, 0) = \vec{b}(0, a_2, a_3, 1)$ and $\vec{b}(1, a_2, 0, a_4) = \vec{b}(1, a_2, 1, a_4)$. It is not *a priori* obvious that a gadget with a witness function of this form exists, but we assume for the moment that such a gadget does exist. Following the proof of Lemma 3.7, we note that the size of the domain of a restricted witness function is now 4 (instead of 8). Thus if an $\alpha$-gadget with constant 0 and the restricted witness function exists, an $\alpha$-gadget with constant 0 and at most $2^4$ auxiliary variables exists. Noting that we can identify auxiliary variables identical to $a_1$ or $a_2$, we can consider gadgets with at most 14 auxiliary variables. We can then create a linear program with $\binom{18}{2} = 153$ variables and $2^{18-1} + 8 = 131,080$ constraints. The result of the linear program is the following.

**Lemma 4.4** There exist 8-gadgets with constant 0 reducing $RMBC_{00}$ and $RMBC_{01}$ to CUT. There exist 9-gadgets reducing $RMBC_{10}$ and $RMBC_{11}$ with constant 0 to CUT. All are optimal and strict.

Notice that since we used a restricted witness function, the linear program does not prove that the gadgets are optimal. In order to prove the optimality of our gadget we ran a different linear program. This time we pick only a subset of the constraints arising from part (2) of the definition of a gadget. We restrict our attention to only four of the accepting configurations. For the case of $RMBC_{00}$, these were 0100, 1001, 1010 and 0111. It is clear that since the linear program arising from this has fewer constraints, its solution provides a lower bound on the performance of the gadget reducing $RMBC_{00}$ to CUT with constant 0. Luckily, it turns out that the lower bound obtained in this way equaled the performance of the gadget, thus providing a proof of optimality. In fact, this "under-constrained" LP also produced a valid gadget (more luck!), so the restricted-witness-function trick was not needed after all.[3]

The two lemmas imply the following theorem.

**Theorem 4.5** For every $\gamma > 0$, MAX CUT is hard to approximate to within $59/60 + \gamma \approx .983$. In particular, MAX CUT is hard to approximate to within 60/61.

[3]The optimum objective function value of a LP is of course unique but in general the corresponding primal (and dual) solutions are not unique. We have observed most or all of our LP's producing different solutions — different optimal gadgets. For this lower-bounding LP, it is possible that a different solution would not happen to produce a gadget.

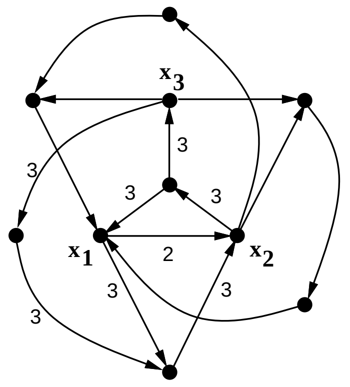

Figure 2: 8-gadget reducing $\mathrm{PC}_0$ to DICUT. Edges have weight 1 except when marked otherwise.

## 4.2 MAX DICUT

An analysis similar to that above leads to linear programs generating gadgets reducing members of PC and RMBC to DICUT. The only difference is that in this case we do not need gadgets with a constant.

**Lemma 4.6** There exist 6.5-gadgets reducing $\mathrm{PC}_0$ and $\mathrm{PC}_1$ to DICUT, and they are optimal and strict.

The $\mathrm{PC}_0$ gadget is shown in Figure 2.

**Lemma 4.7** There exists a 6-gadget reducing $\mathrm{RMBC}_{00}$ to DICUT, 6.5-gadgets reducing $\mathrm{RMBC}_{01}$ and $\mathrm{RMBC}_{10}$ to DICUT and a 7-gadget reducing $\mathrm{RMBC}_{11}$ to DICUT. All are optimal and strict.

**Theorem 4.8** For every $\gamma > 0$, MAX DICUT is hard to approximate to within $131/134 + \gamma \approx .978$. In particular, MAX DICUT is hard to approximate to within $44/45$.

## 4.3 MAX 2-CSP

**Lemma 4.9** There exist 5-gadgets reducing $\mathrm{PC}_0$ and $\mathrm{PC}_1$ to 2CSP, and they are optimal and strict.

**Lemma 4.10** There exist 5-gadgets reducing every constraint function in the family RMBC to 2CSP. All are optimal and strict.

**Theorem 4.11** For every $\gamma > 0$, MAX 2CSP is hard to approximate to within $.97 + \gamma$. In particular, MAX 2CSP is hard to approximate to within $33/34$.

MAX 2CSP can be approximated within .859 [3]. The above theorem has implications for probabilistically checkable proofs. Using the well-known reduction from constraint satisfaction problems to probabilistically checkable proofs, Theorem 4.11 implies that constants $c$ and $s$ exist such that $\mathsf{NP} \subseteq \mathsf{PCP}_{c,s}[\log, 2]$ and $c/s > 34/33$. The previously known gap between the completeness and soundness achievable reading two bits was 74/73 [1].

## 5 Interlude: Methodology

Despite the seeming variety, all gadgets in this paper were computed by a single APL2 program (calling OSL to solve the constructed LP). The source function $f$ is specified explicitly, by a small program. Capitalizing on regularities of the problems of interest, the target family $F$ is specified by an underlying function (e.g. disjunction) applied within all clauses of specified length and symmetries (ordered or unordered; variable complementation permitted or prohibited).

Selected assignments are specified explicitly. They define the extended witness function, from which duplicates (under the specified symmetries) are removed. The witness function is equivalent to the auxiliary variables' identities. Re-iterating previous points, selecting of all *accepting assignments* will produce a gadget, selecting of *all* assignments will produce a strict gadget, and selecting a subset of these assignments will produce a lower bound and — with luck — a valid gadget. Use of a "don't-care" state (in lieu of 0 or 1) in selected assignments guarantees a gadget (if the LP is feasible), but not optimality. To illustrate, the most complex gadget specification was that for the reduction from $\mathrm{RMBC}_{00}$ to CUT, whose input was

$$(,0)(\text{'BU'},0)\ \text{RMBC00 MAKE} \neq (00{*}0)(011{*})(10{*}0)(11{*}1).$$

Evaluating *all* assignments to the primary and auxiliary variables defines the inequality constraints of the LP, while *witness* assignments define the equality constraints. After the LP is constructed and solved, an independent verification step confirms the gadget's validity, performance, and strictness.

Complete run times for the hardest gadgets described in this paper were in the range of a half hour on an RS/6000 workstation, with memory usage of 500MB or so. The easiest half-dozen gadgets can be run on a ThinkPad in seconds.

## 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 $(\beta_1, \beta_2)$-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 the 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 $(\beta_1, \beta_2)$-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** [3] 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 [6] from MAX SAT to MAX 3SAT and using a .931-approximation algorithm for MAX 2SAT due to Feige and Goemans [3], one can obtain a .7704-approximation algorithm for MAX 3SAT. The basic idea of [6] 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 $(\beta_1, \beta_2)$-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 $\alpha$-gadget reducing $f$ to 2SAT, there exists a $(\beta_1, \beta_2)$-approximation algorithm for MAX 2SAT, and $\alpha \geq 1 + \frac{(\beta_1 - \beta_2)}{2(1-\beta_2)}$, then there exists a $\rho$-approximation algorithm for MAX 3SAT with

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

**Lemma 6.4** For every function $f \in$ E3SAT, there exists a strict (and optimal) 3.5-gadget reducing $f$ to 2SAT.

Combining Lemmas 6.2, 6.3 and 6.4 we get a .801-approximation algorithm.

**Theorem 6.5** MAX 3SAT has a polynomial-time .801-approximation algorithm.

## 6.2 MAX 3-CONJ SAT

We now turn to the MAX 3ConjSAT problem.

**Lemma 6.6** If for every $f \in$ 3ConjSAT there exists a strict $(\alpha_1 + \alpha_2)$-gadget reducing $f$ to 2SAT composed of $\alpha_1$ length-1 clauses and $\alpha_2$ length-2 clauses, and there exists a $(\beta_1, \beta_2)$-approximation algorithm for MAX 2SAT, then there exists a $\rho$-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 $\alpha_1 + \alpha_2 > 1 + 1/8(1-\beta_2)$.

The proof is similar to that of Lemma 6.3 and omitted here.

**Lemma 6.7** 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.

**Theorem 6.8** MAX 3ConjSAT has a polynomial-time .367-approximation algorithm.

It is shown by Trevisan [9] 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 $k$ConjSAT. Thus we get the following corollary for PCP systems using 3 bits of queries.

**Corollary 6.9** $\mathsf{PCP}_{c,s}[\log, 3] \subseteq \mathsf{P}$ provided that $c/s > 2.7214$.

The previous best trade-off between completeness and soundness for polynomial-time PCP classes was $c/s > 4$ [9].

## 7 A Lower Bound from Duality

All the computer-constructed gadgets referred to in the preceding sections come with automatic proofs of optimality: the LP formulation guarantees optimality mathematically, and the equality of the objective values computed for the LP and its dual assures optimality in practice. Here we mention one instance where we can use duality to provide lower bounds on the $\alpha$ of a gadget even though such a lower bound can not be constructed on a computer, since the target family is infinite.

**Theorem 7.1** If $\Gamma$ is an $\alpha$-gadget reducing any member of PC to SAT, then $\alpha \geq 4$.

*Proof*: A feasible solution to any LP's dual is a lower bound for the LP. The linear program that finds the best gadget reducing $\mathrm{PC}_0$ to SAT is similar to (LP1), the only difference being that a larger number of clauses are considered, namely, $N = \sum_{i=1}^{7} \binom{7}{i} 2^i$. The dual program is then:

$$\text{maximize} \sum_{\vec{a},\vec{b}:\mathrm{PC}_0(\vec{a})=0} y_{\vec{a},\vec{b}} \qquad \text{(DUAL2)}$$

subject to

$$\begin{array}{lcl} 1+\sum_{\vec{a}:\mathrm{PC}_0(\vec{a})=1} y_{\vec{a},\vec{b}(\vec{a})} & \geq & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \\ \forall j \in [N]: & & \\ \sum_{\vec{a}:\mathrm{PC}_0(\vec{a})=1} \hat{y}_{\vec{a},\vec{b}(\vec{a})} C_j(\vec{a},\vec{b}(\vec{a})) & \leq & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \\ (\forall \vec{a})\ (\forall \vec{b}): \quad y_{\vec{a},\vec{b}} & \geq & 0 \\ (\forall \vec{a}: \mathrm{PC}_0(\vec{a})=1): \quad \hat{y}_{\vec{a},\vec{b}(\vec{a})} & \geq & 0. \end{array}$$

Consider now the following assignment of values to the variables of (DUAL2) (unspecified values are zero):

$$(\forall \vec{a}: \mathrm{PC}_0(\vec{a})=1)\ \hat{y}_{\vec{a},\vec{b}(\vec{a})} = \tfrac{3}{4}$$
$$(\forall \vec{a}: \mathrm{PC}_0(\vec{a})=1)(\forall \vec{a}' : d(\vec{a},\vec{a}')=1)\ y_{\vec{a}',\vec{b}(\vec{a})} = \tfrac{1}{3}$$

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. □

**Note:** In a recent breakthrough result, Hastad [7] has shown that MAX PC is hard to approximate to within $1/2 + \gamma$, for any $\gamma > 0$. The results translate to a surprising threshold of $7/8 + \gamma$ for the approximability of MAX E3SAT. Using the gadgets constructed here, he can also translate this into improved hardness results of $16/17 + \gamma$ and $12/13 + \gamma$ for MAX CUT and MAX DICUT respectively.

**Acknowledgements**

We thank Pierluigi Crescenzi and Oded Goldreich for several crucial discussions and suggestions, and John Forrest and David Jensen for their assistance in efficiently solving large linear programs.

## References

[1] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs and non-approximability – towards tight results (version 3). Technical Report TR95-024, Electronic Colloquium on Computational Complexity, Jan. 1996. See `http://www.eccc.uni-trier.de/eccc/`.

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

[3] 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 Third Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.

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

[5] 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.

[6] 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.

[7] J. Håstad. Unpublished manuscript, 1996.

[8] T. Ono, T. Hirata, and T. Asano. An approximation algorithm for MAX 3SAT. To appear in *Scandinavian Workshop on Algorithm Theory.*, 1996.

[9] L. Trevisan. Positive linear programming, parallel approximation, and PCPs. To appear in *European Symposium on Algorithms*, 1996.

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

# Gadgets, Approximation, and Linear Programming
## [Errata]

Luca Trevisan Gregory B. Sorkin

Madhu Sudan David P. Williamson

We apologize for two errors in the FOCS proceedings version of our paper [2].

The first error was typographical. In the introductory example illustrating a reduction from 3SAT to MAX 2SAT, the 10 clauses replacing $C_k = X_1 \vee X_2 \vee X_3$ should be

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

The second error was in Lemma 3.1 and, as a consequence, Corollary 3.3. However, the further results of the paper all depend on the more specific Lemma 3.5, which is correct, so our principal claims are unaffected.

Lemma 3.1 claimed that for any gadget reducing a constraint $f$ to a constraint family $\mathcal{F}$, there exists an equivalent gadget with at most $K$ auxiliary variables, where $K = K_{f,\mathcal{F}}$ is a finite bound. The error was pointed out by Karloff and Zwick [1], who provide a counterexample in which no finite gadget achieves optimality.

Since the proof of Lemma 3.5 referred to that of Lemma 3.1, we give here a self-contained proof.

**Lemma 3.5** If an $\alpha$-gadget $\Gamma$ reducing $f$ to a hereditary family $\mathcal{F}$ has a witness function for which two auxiliary variables are identical (i.e. $b_{j'}(\cdot) \equiv b_j(\cdot)$), or if an auxiliary variable is identical to a primary variable ($b_{j'}(\vec{a}) \equiv a_j$) then there is an $\alpha'$-gadget $\Gamma'$ using one fewer auxiliary variable, and with $\alpha' \leq \alpha$. If $\Gamma$ is strict, so is $\Gamma'$.

*Proof*: We define a new gadget $\Gamma'$ obtained from $\Gamma$ by replacing each occurrence of $X_{j'}$ by $X_j$ and argue that $\Gamma'$ is an $\alpha'$-gadget reducing $f$ to $\mathcal{F}$ for some $\alpha' \leq \alpha$.

For any constraint $C$ of $\Gamma$, define $red(C)$ as follows. If $X_{j'}$ does not occur in $C$, then $red(C) = C$. Otherwise, we tentatively define $red(C)$ as the constraint obtained from $C$ by replacing the occurrence of $X_{j'}$ by an occurrence of $X_j$. If $C$ did not originally involve $X_j$, then $red(C)$ is a valid constraint from $\mathcal{F}$. If $C$ did involve $X_j$ already, then $red(C)$ contains two occurences of $X_j$, which is not allowed by our definition. However, the hereditary property of $\mathcal{F}$ yields either an equivalent constraint $C' \in \mathcal{F}$ or else the constant function 0 or 1. In this case we reset $red(C)$ to $C'$ or the appropriate constant.

If $\Gamma = (C_1, \ldots, C_m, w_1, \ldots, w_m)$, then define a new gadget $\Gamma' = (red(C_1), \ldots, red(C_m), w_1, \ldots, w_m)$. Correspondingly, let $\vec{b}'(\vec{a})$ be identical to $b(\vec{a})$ but with $b_{j'}$ eliminated. $\Gamma'$ has one fewer auxiliary variable ($X_{j'}$ never occurs in $\Gamma'$).

By construction, $\Gamma'(\vec{a}, \vec{b}'(\vec{a})) \equiv \Gamma(\vec{a}, \vec{b}(\vec{a}))$, so $\Gamma'$ satisfies the gadget-defining equation (2). (Similarly, for strict gadgets $\Gamma$, $\Gamma'$ satisfies (4)). Also, the range of the universal quantification for $\Gamma'$ is smaller than that for $\Gamma$, therefore $\Gamma'$ satisfies inequalities (1) and (3). If constants are produced, subtracting them from both sides of the gadget-defining inequalities (1–4) produces an $(\alpha - w)$-gadget, where $w$ is the total weight on clauses replaced by 1's. $\square$

## Acknowledgements

We thank Howard Karloff and Uri Zwick for bringing both errors to our attention.

## References

[1] H. Karloff and U. Zwick. Personal communication. September 1996.

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