Positive Linear Programming, Parallel Approximation and PCP's*

Luca Trevisan

Università di Roma *La Sapienza*, Dipartimento di Scienze dell'Informazione, Via Salaria 113, I-00198 Roma, Italy. Email: trevisan@dsi.uniroma1.it.

Abstract. Several sequential approximation algorithms are based on the following paradigm: solve a linear or semidefinite programming relaxation, then use randomized rounding to convert fractional solutions of the relaxation into integer solutions for the original combinatorial problem. We demonstrate that such a paradigm can also yield *parallel* approximation algorithms by showing how to convert certain linear programming relaxations into essentially equivalent *positive linear programming* [18] relaxations that can be near-optimally solved in NC. Building on this technique, and finding some new linear programming relaxations, we develop improved parallel approximation algorithms for Max Sat, Max DiCut, and Max k-CSP. We also show a connection between probabilistic proof checking and a restricted version of Max k-CSP. This implies that our approximation algorithm for Max k-CSP can be used to prove inclusion in P for certain PCP classes.

1 Introduction

Several approximation algorithms for combinatorial optimization problems are based on the following paradigm: find a mathematical programming (usually, linear or semidefinite programming) relaxation of the problem, that can be solved in polynomial time, and then prove that any feasible "fractional" solution for the relaxation can be *rounded* to yield a feasible solution for the combinatorial problem whose measure is within a multiplicative factor r from the measure of the original fractional solution. Thus, if we round an optimum solution for the relaxation we will get an r-approximate[1] solution for the combinatorial problem. A well known early example of this technique is Hochbaum's approximation algorithm for Min Weight Vertex Cover [12], where a simple *deterministic* rounding scheme is used. However, *randomized* rounding schemes (first introduced by Raghavan and Thompson [22]) are in general more efficient and are usually derandomizable. For example, Goemans and Williamson [9] use linear

* Research partially supported by the HCM SCOOP project of the European Union. Part of this work was done while the author was visiting the Departament de Llenguatges i Sistemes Informátics of the Universitat Politècnica de Catalunya

[1] We say that a solution is r-approximate ($r < 1$) if its measure is within a multiplicative factor of r from the optimum. See e.g. [6] for formal definitions.

programming and random rounding to give a 3/4-approximation for the MAX SAT problem, matching a previous result by Yannakakis [27] with a simpler algorithm. Outstanding approximation results have been obtained in the last two years by randomly rounding *semidefinite relaxations* of combinatorial problems. Starting with the celebrated results by Goemans and Williamson [10], who showed that MAX CUT and MAX 2SAT are .878-approximable with this technique, an increasing number of results have been obtained using semidefinite programming, including better results for MAX 2SAT and new results for graph coloring, and the "betweeness" problem [8, 14, 7]. Shmoys' recent survey on approximation algorithms [25] contains other applications of linear and semidefinite programming.

Unfortunately, such powerful techniques do not seem to be useful to develop *efficient parallel* approximation algorithms, the main reason being that both linear and semidefinite programming not only are P-hard problems, but it is even P-hard to approximate them [23].

However, there exists a restricted version of linear programming (called Positive Linear Programming, PLP for short) that can be near-optimally solved using an NC algorithm[2] by Luby and Nisan [18]. Luby and Nisan observed that their algorithm could be used to approximate MIN SET COVER in NC within a factor $(1+o(1))\ln\Delta$, were Δ is the maximum cardinality of any set in the family, but, to the best of our knowledge, PLP has never been used to give relaxations of combinatorial problems in combination with random rounding schemes. Indeed, PLP is seemingly a very restricted version of linear programming, capturing *packing* and *covering* problems but nearly nothing else. Contrary to this intuition, we show that some good linear programming relaxations can be "translated" in a PLP form, thus yielding NC approximation algorithms.

The MAX SAT *problem.* We first consider the MAX SAT problem, and its linear programming relaxation due to Goemans and Williamson [9]. In the MAX SAT problem we are given a set $\{C_1, \ldots, C_m\}$ of *disjunctive clauses* over variables $\{x_1, \ldots, x_n\}$ and non-negative weights $w_1, \ldots, w_m$ for the clauses. We seek for an assigment of truth value to the variables $\{x_1, \ldots, x_n\}$ that maximizes the sum of the weights of the satisfied clauses. We show how to convert the linear programming relaxation of MAX SAT of [9] into an "essentially" equivalent PLP relaxation. We also show how to introduce a minor change in Goemans and Williamson's arguments and thus prove the 3/4-approximation guarantee assuming only 5-wise independence. As a consequence, we have that the MAX SAT problem is $(3/4 - o(1))$-approximable in NC. Since PLP can be solved sequentially in quasi-linear time, our translation also implies a $(3/4 - o(1))$-approximate sequential algorithm that runs in $\tilde{O}(m)$, where m is the number of clauses. Recall that the best approximation that is currently achievable for MAX SAT using sequential algorithms [9, 10, 8] is roughly .76, and to obtain

[2] Recall that, informally, an NC algorithm is an algorithm that runs on a parallel computer in poly-logarithmic time using a polynomial number of processors (see e.g. [6] for formal definitions).

such approximation it is necessary to solve semidefinite programs; Yannakakis' 3/4-approximate algorithm [27] requires to solve max flow problems. Our algorithm achieves similar approximation with a remarkably faster running time. The best previous NC approximation for this problem was 1/2, due to Bongiovanni et al. [5] and, independentely, to Hunt et al. [13] using techniques of Luby [17]. More generally, [5, 13] developed NC approximation algorithms for all the problems in the Max SNP [20] class. In particular, their algorithm for MAX SAT requires a quadratic number of processors. More recently, Haglin [11] presented an NC 1/2-approximate algorithm for MAX 2SAT that uses a linear number of processors, and Serna and Xhafa [24] showed that a linear number of processors is sufficient to 1/2-approximate the general MAX SAT problem.

The MAX DICUT *problem.* We then turn to the MAX DICUT problem. In this problem, we are given a directed graph $G = (V, E)$ and npn-negative weights $\{w_{(u,v)}\}_{(u,v)\in E}$ and we search for a partition (V_1, V_2) of the vertices that maximizes the sum of the weights of the edges whose first endpoint is in V_1 and whose second endpoint is in V_2. We give a new linear programming relaxation of such problem and we show that a simple random rounding scheme yields a 1/2-approximation. The linear programming relaxation is then converted into a PLP relaxation. Since the random rounding analysis only required pair-wise independence, we have an NC $(1/2 - o(1))$-approximate algorithm for MAX DICUT. Previous results [17, 5, 13] implied that this problem was 1/4-approximable in NC. Sequential approximation algorithms for this problem are, however, far better: Feige and Goemans [8] (improving a previous .796-approximate algorithm by Goemans and Williamson [10]) recently gave a .855-approximate algorithm using semidefinite programming.

The MAX k-CSP *problem.* In both the above cases, we use PLP relaxations and random rounding to remarkably improve over previous parallel algorithms, but we do not entirely match (or we even fall far below) the performances of known sequential algorithms. Indeed, our results for MAX k-CSP improve even over the best current sequential approximation algorithms. For any $k \geq 1$, the MAX k-CSP problem is the variation of the MAX SAT problem where any clause (also called *constraint*) is allowed to be an arbitrary boolean function over k variables. This problem is somehow implicit in [20] and has been defined in [15] (it has also been called "MAX k FUNCTION SAT" in [2] and "MAX k-GSAT" in [21]). The interest in this problem has been mainly related to the fact that it can express any Max SNP problem. Variations of this problem have also been studied due to their application to the field of Artificial Intelligence (see [16] and the references therein). We show that in order to r-approximate this problem it is sufficient to r-approximate its restricted version MAX kCONJSAT, where each clause is a *conjunction* of literals. For both problems, only 2^{-k}-approximate (see e.g. [15]) algorithms are known. We define a linear programming relaxation of the MAX kCONJSAT problem and we show that a proper random rounding scheme can be used to yield a 2^{1-k} approximation. As in the previous cases, we can then find an equivalent PLP relaxation and obtain an NC $(2^{1-k} - o(1))$-approximate

algorithm. Using our reduction, both algorithms extend to the general MAX k-CSP problem.

Relation to PCP's. Finally, we show that an approximation algorithm for MAX kCONJSAT can be used to *approximate the probability of acceptance of probabilistic verifiers* that *adaptively* read k bits. On the one hand this reduction yields the NP-hardness of approximating MAX kCONJSAT (and thus MAX k-CSP) within a factor $2^{-0.09k}$, on the other hand, together with our improved approximation algorithm, it can be used to show that certain classes of languages defined in terms of probabilistic proof checking are contained in P. Such results strengthen previous ones appeared in [4].

Related and independent results After completing this research, we learnt that Cristina Bazgan independentely used linear programming and random rounding to approximate MAX kCONJSAT within a factor $e/(e^{1/k}+1)^k$ [3]. Such approximation is better than 2^{-k}, but is worse than 2^{1-k}. Motivated by the results of the present paper, a .3674-approximate algorithm for MAX 3 CONJ SAT has been recently developed in [26]. Seemingly, this algorithm does not extend to the general MAX k CONJ SAT problem, and, since it involves semidefinite programming, it cannot be easily parallelized.

Preliminaries

In what follows we will denote by $[n]$ the set $\{1,\ldots,n\}$. Boldface letters will be used to denote vectors, e.g. $\mathbf{u} = (u_1,\ldots,u_m)$. We also use the notations $\tilde{O}(f) \stackrel{\text{def}}{=} O(f(\log f)^{O(1)})$ and $\text{poly}(f) \stackrel{\text{def}}{=} O(f^{O(1)})$. Given an instance I of MAX SAT we let $\mathtt{opt}_{MS}(I)$ be the measure of an optimum solution for I.

Definition 1 [18]. A minimization linear program is said to be an instance of *positive linear programming* (PLP for short) if it is written as

$$\begin{array}{l}\min \mathbf{c}^T\mathbf{x}\\ \text{s.t.}\\ \qquad A\mathbf{x} \geq \mathbf{b}\\ \qquad \mathbf{x} \geq \mathbf{0}\end{array}$$

where all the entries of A, $\mathbf{b}$ and $\mathbf{c}$ are non-negative.

Minimization positive linear programs are also called *covering* problems. Luby and Nisan developed a very efficient algorithm for approximating positive linear programming problems.

Theorem 2 [18]. *There exists a parallel algorithm that given in input a minimization instance P of PLP of size N and a rational $\epsilon > 0$ returns a feasible solution for P whose cost is at most $(1+\epsilon)$ times the optimum. Furthermore, the algorithm runs in time polynomial in $1/\epsilon$ and $\log N$ using $O(N)$ processors.*

The following result is useful to derandomize parallel algorithms where randomization is only needed to generate random variables with limited independence.

Theorem 3 (see e.g. [19], Section 16). *A pairwise independent distribution of n random variables of size $O(n)$ is explicitely constructable in* NC. *For any $k > 2$, a k-wise independent distribution of n random variables of size $O(n^k)$ is explicitely constructable in* NC.

Organization of the paper

Section 2 is devoted to the $(3/4 - o(1))$-approximate algorithm for MAX SAT and Section 3 to the $(1/2 - o(1))$-approximate algorithm for MAX DICUT. The algorithms for MAX k-CSP and the applications to probabilistically checkable proofs are discussed in Sections 4 and 5, respectively. Due to lack of space, some proofs are omitted or sketched.

2 The MAX SAT problem

Let $\{C_1, \ldots, C_m\}$ be a collection of disjunctive boolean clauses over variable set $X = \{x_1, \ldots, x_n\}$ and let $w_1, \ldots, w_m$ be the weights of such clauses. For any clause C_j let us denote by C_j^+ the set of indices of variables occuring positively in C_j and with C_j^- the set of indices of variables occuring negated, so that $C_j = \bigvee_{i \in C_j^+} x_i \vee \bigvee_{i \in C_j^-} \neg x_i$. Let also $w^{\mathrm{tot}} \stackrel{\mathrm{def}}{=} \sum_{j=1}^m w_j$. Goemans and Williamson [9] consider the following linear programming relaxation of the MAX SAT problem.

$$
\begin{array}{ll}
\max \sum_{j=1}^m w_j z_j & \\
\text{s.t.} & \\
\quad z_j \leq \sum_{i \in C_j^+} t_i + \sum_{i \in C_j^-} (1 - t_i) & \text{for all } j \in [m] \qquad \text{(SAT1)} \\
\quad z_j \leq 1 & \text{for all } j \in [m] \\
\quad 0 \leq t_i \leq 1 & \text{for all } i \in [n]
\end{array}
$$

In [9] it is shown that (SAT1) is indeed a relaxation of MAX SAT.

Theorem 4 ([9], Theorem 5.3). *Let $(\mathbf{t}, \mathbf{z})$ be a feasible solution for* (SAT1). *Consider the random assigment such that, for any i, independently, $\mathbf{Pr}[x_i = \mathsf{true}] = \frac{1}{4} + \frac{1}{2} t_i$. Then, for any $j \in [m]$, $\mathbf{Pr}[C_j \text{ is satisfied}] \geq \frac{3}{4} z_j$.*

Starting with an optimum solution for (SAT1), one gets a random assigment that, on the average, has a cost that is at least 3/4 of the optimum. An explicit 3/4-approximate assignment can be found deterministically using the method of conditional expectation ([1], see also [27]).

We shall now show how to convert (SAT1) into an equivalent instance of PLP. The following linear program is clearly equivalent to (SAT1) modulo the substitution of $1 - u_j$ in place of z_j, and the introduction of new variables f_i

that are equal to $1 - t_i$. Also note that additive constants are irrelevant in the objective function, and that we can change the sign of the objective function by changing maximization into minimization.

$$\begin{array}{l} \min \sum_{j=1}^{m} w_j u_j \\ \text{s.t.} \\ \quad \begin{array}{ll} u_j + \sum_{i \in C_j^+} t_i + \sum_{i \in C_j^-} f_i \geq 1 & \text{for all } j \in [m] \\ u_j \geq 0 & \text{for all } j \in [m] \\ t_i + f_i = 1 & \text{for all } i \in [n] \\ t_i, f_i \geq 0 & \text{for all } i \in [n] \end{array} \end{array} \qquad \text{(SAT2)}$$

Fact 1 *For any feasible solution* $(\mathbf{u}, \mathbf{t}, \mathbf{f})$ *for* (SAT2) *of measure* k, $(\mathbf{1} - \mathbf{u}, \mathbf{t})$ *is a feasible solution for* (SAT1) *of measure* $w^{\text{tot}} - k$. *For any feasible solution* $(\mathbf{z}, \mathbf{t})$ *for* (SAT1) *of measure* $w^{\text{tot}} - k$, $(\mathbf{1} - \mathbf{z}, \mathbf{t}, \mathbf{1} - \mathbf{t})$ *is a feasible solution for* (SAT2) *of measure* k.

For any $i \in [n]$, let $occ_i \stackrel{\text{def}}{=} \sum_{i \in C_j^+ \cup C_j^-} w_j$ be the sum of the weights of the clauses where x_i occurrs. Let us consider the following linear program.

$$\begin{array}{l} \min \sum_{j=1}^{m} w_j u_j + \sum_i occ_i (t_i + f_i) \\ \text{s.t.} \\ \quad \begin{array}{ll} u_j + \sum_{i \in C_j^+} t_i + \sum_{i \in C_j^-} f_i \geq 1 & \text{for all } j \in [m] \\ u_j \geq 0 & \text{for all } j \in [m] \\ t_i + f_i \geq 1 & \text{for all } i \in [n] \\ t_i, f_i \geq 0 & \text{for all } i \in [n] \end{array} \end{array} \qquad \text{(SAT3)}$$

The difference between (SAT2) and (SAT3) is that, for any $i \in [n]$, $t_i + f_i$ is allowed to be larger than one, yet this is "discouraged" since $t_i + f_i$ appears in the objective function with a quite large weight. The following lemma formalizes this intuition.

Lemma 5.

1. *Any feasible solution for the* (SAT2) *program of measure* k *is also a feasible solution for the* (SAT3) *program of measure* $k + \sum_i occ_i$.
2. *Given any feasible solution for the* (SAT3) *program of measure* $k + \sum_i occ_i$, *we can compute in* NC *a feasible solution for the* (SAT2) *program of measure at most* k.

Proof. (Sketch) Part (1) is trivial. To prove Part (2), given a feasible solution $(\mathbf{u}, \mathbf{t}, \mathbf{f})$ for (SAT3) whose measure is$k + \sum_i occ_i$ we define a feasible solution $(\mathbf{u}', \mathbf{t}', \mathbf{f}')$ for (SAT2) as follows: for any $i \in [n]$, $t'_i \stackrel{\text{def}}{=} \min\{1, t_i\}$; for any $i \in [n]$, $f'_i \stackrel{\text{def}}{=} 1 - t'_i$; for any $j \in [m]$, $u'_j \stackrel{\text{def}}{=} \max\{u_j, 1 - \sum_{i \in C_j^+} t'_i - \sum_{i \in C_j^-} f'_i\}$.

Note that $(\mathbf{u}', \mathbf{t}', \mathbf{f}')$ can be computed in logarithmic time using a linear number of processors. By tedious but not difficult computations one can show that the measure of $(\mathbf{u}', \mathbf{t}', \mathbf{f}')$ is

$$\sum_{j=1}^{m} w_j u'_j \leq \sum_{j=1}^{m} w_j u_j + \sum_{i=1}^{n} occ_i(t_i + f_i - 1) = k \ .$$

□

We are now ready to prove the main result of this section.

Theorem 6 (Approximation for MAX SAT).

1. *An* RNC *algorithm exists that given an instance of the weighted* MAX SAT *problem and a rational* $\epsilon > 0$*, returns an assigment whose expected measure is at least* $(3/4 - \epsilon)$ *times the optimum. The algorithm runs in* $\mathrm{poly}(1/\epsilon, \log m)$ *time and uses* $O(m+n)$ *processors.*
2. *For any* $\epsilon > 0$*, an* NC $(3/4-\epsilon)$*-approximate algorithm for the weighted* MAX SAT *problem exists that runs in* $\mathrm{poly}(1/\epsilon, \log m)$ *time and uses* $O((m+n)n^5)$ *processors.*
3. *A sequential* $(3/4 - o(1))$*-approximate algorithm for the weighted* MAX SAT *problem exists that runs in* $\tilde{O}(m)$ *time.*

Proof. Let $\phi = (C_1, \ldots, C_m)$ be any instance of MAX SAT, and let $w_1, \ldots, w_m$ be the weights of the clauses. We use the following notation: $J_4 \stackrel{\text{def}}{=} \{j : C_j \text{ contains at most four literals }\}$, $J_5 \stackrel{\text{def}}{=} [m] - J_4$, $\phi_4 \stackrel{\text{def}}{=} \{C_j : j \in J_4\}$, $\phi_5 \stackrel{\text{def}}{=} \{C_j : j \in J_5\}$, $w_4^{\text{tot}} \stackrel{\text{def}}{=} \sum_{j \in J_4} w_j$, and $w_5^{\text{tot}} \stackrel{\text{def}}{=} \sum_{j \in J_5} w_j$. Clearly, we have that $\mathsf{opt}_{MS}(\phi) \leq \mathsf{opt}_{MS}(\phi_4) + w_5^{\text{tot}}$. Let us consider the linear programming relaxations (SAT1), (SAT2) and (SAT3) relative to ϕ_4 and let Z^*_{SAT1}, Z^*_{SAT2} and Z^*_{SAT3}be the optima of these linear programs, respectively. Note that $Z^*_{\text{SAT3}} \leq 5w_4^{\text{tot}}$ (consider e.g. the feasible solution such that $u_j = 1$ for all $j \in [m]$ and $t_i = f_i = 1/2$ for all $i \in [n]$) and that $Z^*_{\text{SAT1}} \geq .5w_4^{\text{tot}}$ (consider e.g. the feasible solution where all variables are equal to 1/2), thus we have that $Z^*_{\text{SAT3}} \leq 10Z^*_{\text{SAT1}}$. Since (SAT3) is an instance of positive linear programming, we can use Theorem 2 to find in NC a solution $(\mathbf{u}, \mathbf{t}, \mathbf{f})$ for (SAT3) whose measure is at most $(1+2\epsilon/15)Z^*_{\text{SAT3}}$. Let $(\mathbf{u}', \mathbf{t}', \mathbf{f}')$ be the corresponding feasible solution for (SAT2) that we can find as stated in Lemma 5, Part (2). It is immediate to see that $(\mathbf{z}, \mathbf{t}') \stackrel{\text{def}}{=} (\mathbf{1} - \mathbf{u}', \mathbf{t}')$ is a feasible solution for (SAT1). The difference between the optimum measure and the measure of such solution is

$$Z^*_{\text{SAT1}} - \sum_{j=1}^{m} w_j z_j = \sum_{j=1}^{m} w_j u'_j - Z^*_{\text{SAT2}}$$

$$\leq \sum_{j=1}^{m} w_j u_j + \sum_{i=1}^{n} occ_i(t_i + f_i) - Z^*_{\text{SAT3}} \leq \frac{2}{15}\epsilon Z^*_{\text{SAT3}} \leq \frac{4}{3}\epsilon Z^*_{\text{SAT1}} \ .$$

Consider now the random assigment such that x_i is true with probabilty $1/4 + t'_1/2$. Note that in such assigment each literal is true with probability at least

1/4, and thus a clause with five or more literals is true with probability at least $1-(3/4)^5 = 0.76269\ldots > 3/4$. From Theorem 4 and from the above considerations we have that the average measure of such assigment is

$$\sum_{j=1}^{m} w_j \mathbf{Pr}[C_j \text{ is satisfied }] \geq \frac{3}{4}\sum_{j\in J_4}^{m} w_j z_j + \sum_{j\in J_5} 0.76269 w_j$$
$$\geq \frac{3}{4}\left(1-\frac{4}{3}\epsilon\right) Z^*_{\mathrm{SAT1}} + \frac{3}{4} w_5^{\mathrm{tot}} \geq \left(\frac{3}{4}-\epsilon\right) \mathsf{opt}_{MS}(\phi) \ .$$

The time bound follows from the fact that the instance of positive linear programming to be solved has size $O(m)$.

To prove Part (2) just note that the above analysis only assumed 5-wise independence. From Theorem 3 we have that a 5-wise independent probability distribution over n random variables exists of size $O(n^5)$. We can thus run in parallel $O(n^5)$ copies of the above algorithm (one for each element of the distribution) and then take the best outcome.

Finally, regarding Part (3), one can use a sequential version of Luby and Nisan's algorithm to approximate the relaxation. Since the size of the relaxation is bounded by m, it will take $O(m(\log m)^{O(1)})$ time to find a $(1+\epsilon)$-approximate solution, provided that $\epsilon = 1/(\log m)^{(O(1))}$. After applying random rounding, derandomization can be done in linear time using conditional expectation (see e.g. [27]). Observe that, while doing the derandomization, we can ignore all literals occuring in a clause but the first five (this is compatible with our approximation analysis). Thus, derandomization can be done in $O(m)$ time, independent of n. □

3 The MAX DICUT problem

Let $G=(V,E)$ be a directed graph with n nodes and m edges (for simplicity of notation we assume that $V=[n]$). Let also $\{w_{(i,j)} : (i,j)\in E\}$ be weights over the edges and define $w^{\mathrm{tot}} \stackrel{\mathrm{def}}{=} \sum_{(i,j)\in E} w_{(i,j)}$. Consider the following linear programming relaxation of the MAX DICUT problem.

$$\begin{array}{ll} \max \sum_{(i,j)\in E} w_{(i,j)} z_{(i,j)} & \\ \text{s.t.} & \\ \quad z_{(i,j)} \leq t_i & \text{for all } (i,j)\in E \\ \quad z_{(i,j)} \leq 1-t_j & \text{for all } (i,j)\in E \\ \quad 0 \leq t_i \leq 1 & \text{for all } i \in V \end{array} \qquad \text{(DI1)}$$

To see that (DI1) is indeed a relaxation of MAX DICUT, note that given a cut (V_1,V_2) we can construct a feasible solution for (DI1) such that each variable is equal either to zero or to one, $t_i=1$ if and only if $i\in V_1$, and $z_{(i,j)}=1$ if and only if $i\in V_1$ and $j\in V_2$. The measure of such solution is equal to the measure of the cut.

We shall now prove an analogous of Theorem 4, that is, we shall show that given any solution for (DI1), we can define a random cut such that the probability of an edge (i,j) to be in the cut is close to $z_{(i,j)}$.

Theorem 7 (Random rounding for (DI1)**).** *Let* $(\mathbf{z},\mathbf{t})$ *be a feasible solution for* (DI1)*, and consider the random cut such that, for any node* i*,* $\mathbf{Pr}[i \in V_1] = \frac{1}{4} + \frac{t_i}{2}$*. Then for any edge* $(i,j) \in E$*, the probability of the edge to be in the cut is at least* $z_{(i,j)}/2$*.*

Proof. Note that $\mathbf{Pr}[i \in V_2] = 1 - \left(\frac{1}{4} + \frac{t_i}{2}\right) = \frac{1}{4} + \frac{1-t_i}{2}$. Then, we have that

$$\mathbf{Pr}[i \in V_1 \text{ and } j \in V_2] = \left(\frac{1}{4} + \frac{t_i}{2}\right)\left(\frac{1}{4} + \frac{1-t_j}{2}\right) \geq \left(\frac{1}{4} + \frac{1}{2}z_{(i,j)}\right)^2 \geq \frac{1}{2}z_{(i,j)} \ .$$

Where first inequality follows from the constraints on $z_{(i,j)}$. □

Remark. The above analysis is tight, as can be shown by considering the directed complete graph with $2n$ vertices and $2n(2n-1) = 4n^2 - 2n$ edges (assume that all weights are one). Then the optimum of the MAX DICUT problem is clearly n^2 (the balanced partition), while the solution with all variables equal to $1/2$ is feasible for (DI1) and has measure $2n^2 - n$. The ratio between the two values is arbitrarily close to $1/2$.

Note also that the above analysis only assumed pairwise independence. As in the preceding section, we can convert (DI1) into an equivalent instance of PLP.

$$\begin{array}{l}
\min \sum_{(i,j)\in E} w_{(i,j)}(u_{(i,j)} + p_i + q_i + p_j + q_j) \\
\text{s.t.} \\
\qquad \begin{array}{ll}
u_{(i,j)} + t_i \geq 1 & \text{for all } (i,j) \in E \\
u_{(i,j)} + f_j \geq 1 & \text{for all } (i,j) \in E \\
t_i + f_i \geq 1 & \text{for all } i \in V \\
u_{(i,j)} \geq 0 & \text{for all } (i,j) \in E \\
t_i, f_i \geq 0 & \text{for all } i \in V
\end{array}
\end{array} \qquad \text{(DI2)}$$

The following results can then be proved by straightforward modifications of the argument used in the preceding section.

Lemma 8.

1. *For any feasible solution* $(\mathbf{z},\mathbf{t})$ *of* (DI1) *of measure* k*,* $(\mathbf{1}-\mathbf{z},\mathbf{t},\mathbf{1}-\mathbf{t})$ *is a feasible solution of* (DI2) *of measure* $3w^{\mathrm{tot}} - k$*.*
2. *Given a feasible solution* $(\mathbf{u},\mathbf{t},\mathbf{f})$ *of* (DI2) *of measure* $3w^{\mathrm{tot}} - k$ *we can construct in constant parallel time and with* $O(m)$ *processors a feasible solution* $(\mathbf{z},\mathbf{t}')$ *of* (DI1) *of measure at least* k*.*

Theorem 9 (Approximation for MAX DICUT**).**

1. *An* RNC *algorithm exists that given an instance of the weighted* MAX DICUT *problem and a rational* $\epsilon > 0$*, returns a cut whose expected measure is at least* $(1/2 - \epsilon)$ *times the optimum. The algorithm runs in* $\mathrm{poly}(1/\epsilon, \log m)$ *time and uses* $O(m+n)$ *processors.*
2. *For any* $\epsilon > 0$*, an* NC $(1/2-\epsilon)$*-approximate algorithm for the weighted* MAX DICUT *problem exists that runs in* $\mathrm{poly}(1/\epsilon, \log m)$ *time and uses* $O((m+n)n)$ *processors.*

4 The MAX *k*-CSP problem

In this section we deal with the MAX k-CSP problem. We begin by showing that, without loss of generality, we can restrict ourselves to the simpler MAX kCONJSAT problem.

Theorem 10 (MAX k-CSP vs. MAX kCONJSAT). *For any* $k > 1$ *and for any* r*,* $0 < r \leq 1$*, if* MAX kCONJSAT *is* r*-approximable, then* MAX k-CSP *is* r*-approximable.*

Proof. (Sketch) We can reduce an instance ϕ_{CSP} of the MAX k-CSP problem to an instance ϕ_{CSAT} of the MAX kCONJSAT problem by substituting every constraint of ϕ_{CSP} with the set of conjunctive clauses that occur in its DNF expression. It is easy to show that any solution satisfying s constraints in ϕ_{CSP} will satisfy s clauses in ϕ_{CSAT}, and vice versa. □

We shall now prove that, for any $k \geq 1$, MAX kCONJSAT is 2^{1-k}-approximable. As in the preceding sections, we shall give a linear programming relaxation of the problem and a proper randomized rounding scheme. Assume we have an instance of MAX kCONJSAT given by constraints $C_1, \ldots, C_m$, whose weights are $w_1, \ldots, w_m$ over variables $x_1, \ldots, x_n$. The linear programming relaxation has a variable t_i for any variable x_i of the MAX kCONJSAT problem, plus a variable z_j for any constraint C_j. Furthermore, we denote by C_j^+ (respectively, C_j^-) the set of indices of positive (respectively, negative) literals in C_j, so that $C_j = \bigwedge_{i \in C_j^+} x_i \ \wedge \ \bigwedge_{i \in C_j^-} \neg x_i$. The formulation is

$$
\begin{array}{ll}
\max \sum_j w_j z_j & \\
\text{s.t.} & \\
\quad z_j \leq t_i & \text{for all } j \in [m], i \in C_j^+ \qquad \text{(CSP)} \\
\quad z_j \leq 1 - t_i & \text{for all } j \in [m], i \in C_j^- \\
\quad 0 \leq t_i \leq 1 & \text{for all } i \in [n]
\end{array}
$$

The proof that (CSP) is a relaxation of MAX kCONJSAT is identical to the proof that (SAT1) is a relaxation of MAX SAT.

Theorem 11 (Random rounding for (CSP)). *Let* $(\mathbf{z}, \mathbf{t})$ *be a feasible solution for* (CSP)*, consider the random assigment such that* $\mathbf{Pr}[x_i = \mathsf{true}] = \frac{k-1}{2k} + \frac{t_i}{k}$*. Then, for any clause* C_j*, the probability that it is satisfied by the random assigment is at least* $\frac{z_j}{2^{k-1}}$*.*

Proof. Note that, according to the random assigment, $\mathbf{Pr}[x_i = \mathsf{false}] = \frac{k-1}{2k} + \frac{1-t_i}{k}$. Let us assume that C_j is a h-ary constraint for some $h \leq k$.

$$\begin{aligned}\mathbf{Pr}[C_j \text{ is satisfied }] &= \left(\prod_{i \in C_j^+} \frac{k-1}{2k} + \frac{t_i}{k}\right) \cdot \left(\prod_{i \in C_j^-} \frac{k-1}{2k} + \frac{1-t_i}{k}\right) \\ &\geq \left(\frac{k-1}{2k} + \frac{z_j}{k}\right)^h \geq \left(\frac{k-1}{2k} + \frac{z_j}{k}\right)^k \geq \frac{z_j}{2^{k-1}} \ .\end{aligned}$$

Where first inequality follows from the constraints on z_j, second inequality from the fact that $h \leq k$ and the last inequality can be proved by studying the first derivative of the function

$$f(z) = \frac{(\frac{k-1}{2k} + \frac{z}{k})^k}{z}$$

and showing that in the interval $(0, 1)$ it reaches its minimum for $z = 1/2$: in that point we have that $f(1/2) = 2^{1-k}$. □

Remark. The above analysis is tight, as can be shown by considering an instance $C_1, \ldots, C_{2^k}$ where the clauses are all the possible size-k conjunction of $\{x_1, \ldots, x_k\}$. Any assigment to $\{x_1, \ldots, x_k\}$ will satisfy exactly one clause (that is, the optimum is equal to 1). On the other hand, the feasible solution for (CSP) where all variables are equal to $1/2$ has measure 2^{k-1}.

Theorem 12 (Approximation for MAX kCONJSAT**).** *For any $k \geq 1$, the weighted* MAX kCONJSAT *problem is 2^{1-k}-approximable in polynomial time, and is $(2^{1-k} - o(1))$-approximable in* NC.

Proof. Regarding the first claim, in order to compute a 2^{1-k}-approximate solution it is sufficient to optimally solve (CSP), then use the random rounding scheme described in Theorem 11 and finally use conditional expectation (see [1]) to obtain an assigment whose measure is no smaller than the average measure of such random assigment. The approximation guarantee follows from Theorem 11. To prove the second claim, one has to rewrite (CSP) as a positive linear program (as done in the preceding sections) near optimally solve it with Luby and Nisan's algorithm, and use k-wise independent distributions to do derandomization. □

Corollary 13 (Approximation for MAX k-CSP**).** *For any $k > 1$, the weighted* MAX k-CSP *problem is 2^{1-k}-approximable in polynomial time, and $(2^{1-k} - o(1))$-approximable in* NC.

5 Relations with Proof Checking

We start giving some definitions about probabilistically checkable proofs (we follow the notation used in [4]). A *verifier* is an oracle probabilistic polynomial-time Turing machine V. During its computation, V tosses random coins, reads its input and has oracle access to a string π called *proof*. Let x be an input and π be a proof. We denote by $\mathbf{ACC}[V^\pi(x)]$ the probability over its random tosses that V accepts x using π as an oracle. We also denote by $\mathbf{ACC}[V(x)]$ the maximum of $\mathbf{ACC}[V^\pi(x)]$ over all proofs π. The efficiency of the verifier is determined by several parameters. In particular, if V is a verifier and L is a language, we say that

- V *uses* $r(n)$ *random bits* (where $r : \mathcal{Z}^+ \to \mathcal{Z}^+$ is an integer function) if for any input x and for any proof π, V tosses at most $r(|x|)$ random coins;
- V *has query complexity* q (where q is an integer) if for any input x, any random string R, and any proof π, V reads at most q bits from π;
- V *has soundness* s (where $s \in [0,1]$ is a real) if, for any $x \notin L$, $\mathbf{ACC}[V(x)] \leq s$;
- V *has completeness* c (where $c \in [0,1]$ is a real) if, for any $x \in L$, $\mathbf{ACC}[V(x)] \geq c$.

Remark. Note that a verifier that has query complexity q can read its q bits *adaptively*, that is, the i-th access to the proof may depend on the outcomes of the previous $i-1$ accesses.

Definition 14 (PCP classes). Let L be a language, let $0 < s < c \leq 1$ be any constants, q be a positive integer and $r : \mathcal{Z}^+ \to \mathcal{Z}^+$, then we say that $L \in \mathsf{PCP}c, sr, q$ if a verifier V exists for L that uses $O(r(n))$ random bits, has query complexity q, soundness s and completeness c.

Several recent results about the hardness of approximation of combinatorial problems (including Max Sat [4] and Max DiCut [4, 26]) have been proved using the fact, proved in [4], that $\mathsf{NP} = \mathsf{PCP}1, s\log, 3$ for any $s > 0.85$. The verifier developed to prove such result is adaptive. Using less than 3 queries or having a soundness smaller than 0.85 would immediately imply improved non-approximability results. Due to such consideration, it seems interesting to consider what kind of combinations of parameters may be sufficient to characterize NP, and which one are too weak (unless $\mathsf{P} = \mathsf{NP}$). The next result implies that one can prove inclusion of PCP classes into P by simply developing approximation algoritms for Max kConjSAT.

Theorem 15 (Max kConjSAT vs PCP). *If*
Max kConjSAT *is* r*-approximable for some* $r \leq 1$*, then* $\mathsf{PCP}c, s\log, k \subseteq \mathsf{P}$ *for any* $c/s > 1/r$.

Proof. (Rough Sketch) We assume familiarity with the terminology of [4]. For any of the $2^{O(\log n)} = \mathrm{poly}(n)$ possible random strings we consider the behaviour

of the verifier and we encode it using a $(1,0)$-*gadget*. Such gadget contains a conjunctive clause for each *accepting configuration* of the verifier. One should note that even if the verifier (being adaptive) can read up to $2^k - 1$ bits, only k bits are specified in any accepting configuration. Thus the gadget contains only k Conj SAT clauses. At this point, the theorem follows using standard calculations. □

A first consequence of Theorem 15 is that Max kConjSAT is hard to approximate even within very small factors.

Theorem 16 (Hardness of Max kConjSAT). *For any $k \geq 11$, if* Max kConjSAT *is $2^{-\lfloor k/11 \rfloor}$-approximable, then* $\mathsf{P} = \mathsf{NP}$.

Proof. Bellare Goldreich and Sudan [4] prove that $\mathsf{NP} = \mathsf{PCP}1, 0.5\log, 11$. Then, $\lfloor k/11 \rfloor$ independent repetitions of their protocol yield $\mathsf{NP} = \mathsf{PCP}1, 2^{-\lfloor k/11 \rfloor}\log, k$: applying Theorem 15, the claim follows. □

The following result can be obtained combining Theorems 12 and 15

Theorem 17 (Weak PCP classes). $\mathsf{PCP}c, s\log, q \subseteq \mathsf{P}$ *for any $c/s > 2^{q-1}$. In particular,* $\mathsf{PCP}1, 0.249\log, 3 \subseteq \mathsf{P}$.

The above theorem improves over previous results by Bellare, Goldreich and Sudan [4], stating that $\mathsf{PCP}c, s\log, q \subseteq \mathsf{P}$ for any $c/s > 2^q$ and $\mathsf{PCP}1, 0.18\log, 3 \subseteq \mathsf{P}$, respectively.

Acknowledgments

I wish to thank Pierluigi Crescenzi for suggesting the problem, encouraging this research, and giving several useful suggestions. I am also grateful with Madhu Sudan and Fatos Xhafa for helpful comments on a preliminary version of this paper.

References

1. N. Alon and J. Spencer. *The Probabilistic Method.* Wiley Interscience, 1992.
2. S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of graph problems. In *Proc. of the 27th ACM Symposium on Theory of Computing*, pages 284–293, 1995.
3. M. Bellare. Proof checking and approximation: Towards tight results. *Sigact News*, 27(1), 1996.
4. M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results (3rd version). Technical Report ECCC TR95-24, 1995. Preliminary version in *Proc. of FOCS'95.*
5. G. Bongiovanni, P. Crescenzi, and S. De Agostino. Descriptive complexity and parallel approximation of optimization problems. Manuscript, 1991.

6. D.P. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1993.
7. B. Chor and M. Sudan. A geometric approach to betweennes. In *Proc. of the 3rd European Symposium on Algorithms*, 1995.
8. U. Feige and M. Goemans. Approximating the value of two provers proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proc. of the 3rd IEEE Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.
9. M. Goemans and D. Williamson. New $3/4$-approximation algorithms for the maximum satisfiability problem. *SIAM J. on Discrete Mathematics*, 7(4):656–666, 1994. Preliminary version in *Proc. of IPCO'93*.
10. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. of the ACM*, 42(6):1115–1145, 1995. Preliminary version in *Proc. of STOC'94*.
11. D.J. Haglin. Approximating maximum 2-CNF satisfiability. *Parallel Processing Letters*, 2:181–187, 1992.
12. D. Hochbaum. Approximation algorithms for set covering and vertex cover problems. *SIAM J. on Computing*, 11:555–556, 1982.
13. H. B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, and R.E. Stearns. Every problem in MAX SNP has a parallel approximation algorithm. Manuscript, 1993.
14. D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *Proc. of the 35th IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1994.
15. S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. In *Proc. of the 35th IEEE Symposium on Foundations of Computer Science*, pages 819–830, 1994.
16. H.C. Lau and O. Watanabe. Randomized approximation of the constraint satisfaction problem. In *Proc. of the 5th Scandinavian Workshop on Algorithm Theory*, 1996.
17. M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. on Computing*, 15:1036–1053, 1986.
18. M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. of the 25th ACM Symposium on Theory of Computing*, pages 448–457, 1993.
19. M. Luby and A. Wigderson. Pairwise independence and derandomization. Technical Report ICSI TR-95-035, 1995.
20. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*.
21. C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
22. P. Raghavan and C.D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
23. M. Serna. Approximating linear programming is log-space complete for P. *Information Processing Letters*, 37, 1991.
24. M. Serna and F. Xhafa. On parallel versus sequential approximation. In *Proc. of the 3rd European Symposium on Algorithms*, pages 409–419, 1995.
25. D. Shmoys. Computing near-optimal solutions to combinatorial optimization problems. In *Combinatorial Optimization*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 20, 1995.

26. 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*, 1996.
27. M. Yannakakis. On the approximation of maximum satisfiability. *J. of Algorithms*, 17:475–502, 1994. Preliminary version in *Proc. of SODA'92*.

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