A Combinatorial Algorithm for MAX CSP

MAYUR DATAR* TOMÁS FEDER† ARISTIDES GIONIS‡
RAJEEV MOTWANI§ RINA PANIGRAHY¶

September 13, 2002

Abstract

We consider the problem MAX CSP over multi-valued domains with variables ranging over sets of size $s_i \leq s$ and constraints involving $k_j \leq k$ variables. We study two algorithms with approximation ratios A and B, respectively, so we obtain a solution with approximation ratio $\max(A, B)$.

The first algorithm is based on the linear programming algorithm of Serna, Trevisan, and Xhafa [12] and gives ratio A which is bounded below by s^{1-k}. For k=2, our bound in terms of the individual set sizes is the minimum over all constraints involving two variables of $(\frac{1}{2\sqrt{s_1}} + \frac{1}{2\sqrt{s_2}})^2$, where s_1 and s_2 are the set sizes for the two variables.

We then give a simple combinatorial algorithm which has approximation ratio B, with $B > A/e$. The bound is greater than s^{1-k}/e in general, and greater than $s^{1-k}(1-\frac{s-1}{2(k-1)})$ for $s \leq k-1$, thus close to the s^{1-k} linear programming bound for large k. For $k = 2$, the bound is $\frac{4}{9}$ if $s = 2$, $\frac{1}{2(s-1)}$ if $s \geq 3$, and in general greater than the minimum of $\frac{1}{4s_1} + \frac{1}{4s_2}$ over constraints with set sizes s_1 and s_2, thus within a factor of two of the linear programming bound.

For the case of $k = 2$ and $s = 2$ we prove an integrality gap of $\frac{4}{9}(1 + O(n^{-\frac{1}{2}}))$. This shows that our analysis is tight for any method that uses the linear programming upper bound.

Keywords: Algorithmical approximation, analysis of algorithms, combinatorial problems, databases, design of algorithms, graph algorithms.

*Department of Computer Science, Gates 4B, Stanford University, Stanford, CA 94305-9045. E-mail: `datar@cs.stanford.edu`. Supported by NSF Grant IIS-0118173 and a Microsoft Research Graduate Fellowship.

†E-mail: `tomas@theory.stanford.edu`.

‡Department of Computer Science, Gates 4B, Stanford University, Stanford, CA 94305-9045. E-mail: `gionis@cs.stanford.edu`. Supported by NSF Grant IIS-0118173 and a Microsoft Research Graduate Fellowship.

§Department of Computer Science, Gates 4B, Stanford University, Stanford, CA 94305-9045. E-mail: `rajeev@cs.stanford.edu`. Supported by NSF Grant IIS-0118173 and an Okawa Foundation Research Grant.

¶E-mail: `rinap@cisco.com`.

1 Introduction

We consider the problem MAX CSP (Maximization Constraint Satisfaction Problem) over multi-value domains [12]. An instance of this problem consists of n variables x_i, where each variable takes values from a corresponding set S_i. Each set S_i has $s_i = |S_i|$ values, and we define s to be the largest set size i.e. $\max_i s_i$. The problem statement also specifies a set of m constraints, where each constraint is defined in the following way: For the j-th constraint, a set of indices $A_j \subseteq [n]$ specifies which variables participate in it. The number of variables in each constraint can be arbitrary, so we denote by k_j the cardinality of the set $A_j = \{i_1, \ldots, i_{k_j}\}$. We also define k to be $\max_j k_j$. We write $(x_i | i \in A_j)$ to denote the k_j-tuple of variables that participate in the j-th constraint. Then, the j-th constraint is defined as a nonempty relation $R_j \subseteq T_j = \prod_{i \in A_j} S_i$, and a weight $w_j \geq 0$ is associated with it. The relation R_j can be interpreted as an enumeration of all the valid assignments to the variables $(x_i | i \in A_j)$ (from the set of possible assignments $T_j = \prod_{i \in A_j} S_i$) that make the j-th constraint satisfied. The objective of MAX CSP is to assign a value to each variable x_i from the corresponding S_i so as to maximize the sum of weights of satisfied relations, i.e. the sum of the w_j such that $(x_i | i \in A_j) \in R_j$.

This problem is a common generalization of several known and well-studied problems. Multi-valued variables is a natural generalization of boolean variables. The special cases of MAX CSP that have been studied in the literature usually have the same number of values per variable, i.e. $s_i = s$ for all $i \in [n]$, and/or the same number of variables per constraint, i.e. $k_j = k$ for all $j \in [m]$.

To give a few concrete examples, MAX CSP is a natural generalization of the boolean constraint satisfaction problem MAX k-CSP. It also generalizes the well-studied MAX CUT and MAX DICUT problems for which algorithms based on semidefinite programming have been proposed ([6, 8]). The reductions for MAX CUT and MAX DICUT, shown in Figure 1, use $k = 2$ and $s = 2$. Each vertex u in the input graph is replaced with a set $U = \{u_S, u_T\}$, such that the selection of the element in U specifies the part of the cut (S or T) to which the corresponding vertex will be assigned. For MAX CUT, for each edge (u, v) in the input graph, we include in the MAX CSP instance the relation $R_{uv} = \{(u_S, v_T), (u_T, v_S)\}$ with weight $w_{uv} = 1$. For MAX DICUT, for each directed edge (u, v) we include the relation $R_{uv} = \{(u_S, v_T)\}$ with weight $w_{uv} = 1$. Serna, Trevisan, and Xhafa [12] discuss several other special cases.

Besides being of theoretical importance as a generalization of many well-studied problems, our interest in this problem is motivated by an application in databases. This application is that of physical database design, as for example discussed by Chaudhuri and Narasayya [4] in the context of index-selection for Microsoft SQL server (see also [1]). Given a database consisting of a set of relations and a workload (set of queries), the goal is to choose for each relation a clustered index (the order in which the relation is stored on disk). The choice of the relation's attribute on which to create a clustered index can be modeled by a variable which takes on exactly one of the values from a domain (the set of possible attributes to index in the relation). Each query in the workload obtains a certain benefit if the set of clustered indexes pertinent to the query are all selected. Thus, we can model the benefit for a query by a constraint over the clustered indexes. The problem of choosing the clustered indexes so as to maximize the sum of the benefits for the queries in the workload is then

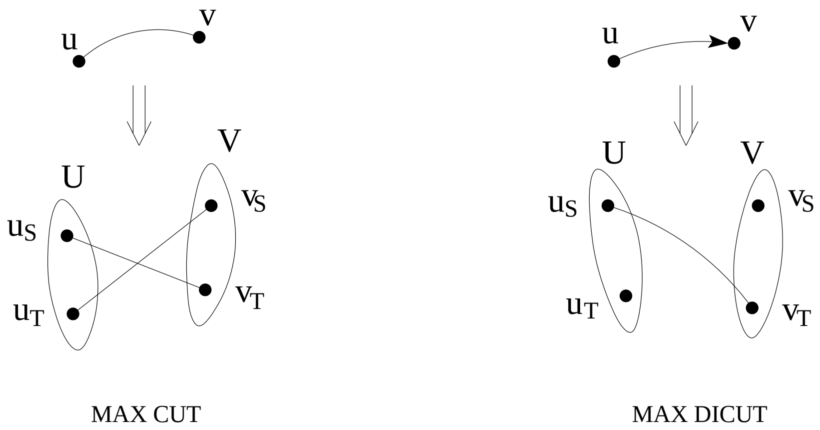

Figure 1: Reductions of MAX CUT and MAX DICUT to MAX CSP

exactly the MAX CSP problem over multi-valued domains.

If we are to design approximations algorithms for applications such as the one presented above, it precludes the use of linear programming or semidefinite programming techniques for the sake of efficiency. In other words, algorithms that are purely combinatorial are preferred for a real implementation. One of the main contributions of this paper is a simple combinatorial algorithm that achieves an approximation ratio very close (within factor e) to that of the best-known algorithm (which uses linear programming).

Related Work: There is a simple algorithm (see Section 2) for this problem that achieves an approximation ratio of $\min_j |R_j|/|T_j|$. When all $k_j = 2$ and all $s_i = 2$, the best known approximation ratios for the cases where $\min_j |R_j|/|T_j| = 1/4, 2/4,$ and $3/4$ are $0.859, 0.878,$ and 0.931, respectively, by semidefinite programming [6, 8]; these three results apply when some additional R_j with $k_j = 1$ are included as well. On the other hand, the semidefinite programming method does not extend in a straightforward manner to arbitrary-sized domains; the structure of the specific problem at hand need to be studied in detail. Successful generalizations of the semidefinite programming method have used very different modeling techniques, such as *(i)* associating each variable with one unit vector and rounding using many random vectors (used by Frieze and Jerrum [7] for MAX k-CUT), *(ii)* associating each variable with many unit vectors (used by Anderson et al [2] for MAX 2-LIN MOD p), or *(iii)* associating each variable with a unit vector in the complex space (used by Goemans and Williamson [9] for MAX 3-DICUT and for improved bounds on MAX 3-CUT and MAX 2-LIN MOD 3). Here MAX k-LIN MOD p denotes the problem of maximizing the number of linear equation mod p, with at most k variables per equation.

In the rest of this section we summarize our results and we discuss their relation with previously known ones.

Linear programming algorithm: we first consider the linear programming formulation (LP1) of Serna, Trevisan, and Xhafa [12]. We follow their proof ideas but we give a more detailed analysis for the general problem, where the set sizes s_i and the number of variables k_j are arbitrary. We show that the linear programming achieves an approximation ratio of $A = \min_j |T_j|^{\frac{1}{k}-1}$, where $k = \max_j k_j$. In particular, if all $s_i = s$, we obtain as a special case the approximation ratio of s^{1-k} shown by Serna et al [12]. More specifically, this gives a ratio of $1/s$ for the case where $k = 2$. When $k = 2$ but the set sizes s_i are arbitrary the approximation ratio of the linear programming algorithm can be improved to $\min_{j:T_j=S_p\times S_q}\left(\frac{1}{2\sqrt{s_p}}+\frac{1}{2\sqrt{s_q}}\right)^2$.

Combinatorial algorithm: We then give a simpler, combinatorial, algorithm. The idea is to consider a new linear programming formulation (LP2) which is obtained from (LP1) by replacing all the inequalities $y_j \le x_i$ for variable y_j with an equality of the form $y_j = \sum_i \alpha_i x_i$, for appropriate α_j. The new linear program (LP2) can be analyzed to provide guarantees similar to those of (LP1). Moreover, it can also be solved trivially and hence the combinatorial algorithm. For the approximation guarantee, if we let $B_j = |T_j|^{-1}\frac{k-1+s_{j\text{tot}}}{k}(1+\frac{1}{k-1})^{1-k_j}$, with $s_{j\text{tot}} = \sum_{i\in A_j} s_i$, the combinatorial algorithm achieves ratio $B = \min_j B_j$. Note that $\frac{k-1+s_{j\text{tot}}}{k} \ge |T_j|^{\frac{1}{k}}$ and $(1+\frac{1}{k-1})^{1-k_j} > 1/e$. Thus the ratio achieved by the combinatorial algorithm is at least $(1/e)\min_j |T_j|^{\frac{1}{k}-1}$. On the other hand, using this result the approximation ratio of (LP1) can be improved to $\min_j \max(|T_j|^{\frac{1}{k}-1}, B_j)$.

When $k = 2$, the ratio achieved is $\frac{4}{9}$ for $s = 2$, $\frac{1}{2(s-1)}$ for $s \ge 3$, and in general greater than $\min_{j:T_j=S_p\times S_q}\frac{1}{4s_p}+\frac{1}{4s_q}$, hence within a factor of two of the bound for the linear programming algorithm. For the case of $k = 2$ and $s = 2$ (MAX DICUT) we prove an integrality gap of $\frac{4}{9}(1+O(n^{-\frac{1}{2}}))$. This shows that our analysis is tight for any method that uses the linear programming upper bound. Needless to say, this lower bound can be overcome by algorithms whose analysis does not use the upper bound obtained from the above linear program. Indeed, Halperin and Zwick [10] recently gave several combinatorial algorithms for MAX DICUT two of which have approximation ratios of $\frac{9}{20}$ and $\frac{1}{2}$, respectively.

The best known hardness result for the case $k = 2$ has instances with $n \le 2s$ and shows that there exists some $\delta > 0$ such that polynomial time approximation to within a factor of $2^{\log^\delta s}$ of the optimal implies that NP has quasipolynomial time deterministic simulations [3]. A hardness bound of $s^{-\lfloor\frac{k}{3}\rfloor}$ for $k \ge 3$ follows from Håstad's work [11]. A hardness bound of the form $s^{-k+O(\sqrt{k})}$ was obtained by Engebretsen [5].

2 Preliminaries

Suppose we have probability distributions for each S_i, and we consider the algorithm ALG_0 that chooses each x_i independently from the corresponding distribution on S_i.

Lemma 1 *Consider a randomized algorithm that picks one value each from every S_i according to a certain probability distribution defined for each S_i. There is a deterministic polynomial time algorithm that obtains a solution (chooses one value from each S_i) whose total weight (objective function) is at least the expected total weight of this randomized algorithm. Furthermore, the running time of the derandomization is $O(\sum_{j=1}^m |R_j| k_j)$.*

Proof: We use the standard technique of conditional probabilities. Select a variable x_1 and the corresponding set S_1. Replace the probabilities in S_1 with variables $y_1, \dots, y_{s_1}$.

The expected total weight of the solution obtained by the randomized algorithm is a linear function f of the y_l.

Select a y_l whose coefficient in f is largest, and deterministically assign x_1 to the value l in S_1. Clearly, the modified distribution so obtained gives a solution of expected total weight at least as large as the expected total weight for the original distribution on S_1. Repeat the process for the remaining variables $x_2, x_3, \ldots$ and corresponding sets $S_2, S_3, \ldots$, one at a time. In the end we obtain the desired solution, deterministically.

For the running time of the above algorithm, notice that each set value is considered only once, and for a particular value the algorithm examines and updates the weight of all the constraints that contain that value. Thus, the total amount of work is $O(\sum_{i=1}^{n} s_i + \sum_{j=1}^{m} |R_j|k_j)$. Since we don't need to consider set values that do not participate to any constraint, the running time is $O(\sum_{j=1}^{m} |R_j|k_j)$. ■

A simple deterministic algorithm follows from this Lemma.

Corollary 1 *There is a deterministic algorithm that guarantees a ratio of* $\min_j |R_j|/|T_j|$.

Proof: Consider the uniform distribution for x_i over S_i, with $|S_i| = s_i$. The probability of each $(x_{i_1}, \ldots, x_{i_{k_j}}) \in R_j$ is then $\left(\prod_{i \in A_j} s_i\right)^{-1} = 1/|T_j|$. The probability that R_j will be satisfied is thus $|R_j|/|T_j|$.

The expected total weight of the solution thus obtained is therefore

$$\sum_j w_j |R_j|/|T_j| \geq (\min_j |R_j|/|T_j|)(\sum_j w_j).$$

By the preceding lemma, this algorithm can be derandomized and guarantees the ratio $\min_j |R_j|/|T_j|$ since $\sum_j w_j$ is a trivial upper bound for the weight of the optimal. ■

Note, for comparison purposes, that in the case $k = 2$ for linear equations modulo p with two variables per equation, which have $|R_j|/|T_j| = 1/p$, the best-known ratio [2] using semidefinite programming is $1/(p - \Theta(p^{-12}))$.

3 Linear Programming Algorithm

For the results in this section, it is convenient to assume that $|R_j| = 1$ for each j. This can be achieved by replacing an R_j with $|R_j| = r_j$ with r_j distinct relations, one for each element of R_j, and with each of these r_j relations having the same weight w_j of the original R_j.

We consider the linear programming relaxation of the problem by Serna, Trevisan, and Xhafa [12]. For each $S_i = \{a_{i1}, \ldots, a_{is_i}\}$ introduce variables $x_{il} \geq 0$ with $1 \leq l \leq s_i$ and a constraint

$$\sum_{l=1}^{s_i} x_{il} = 1 \tag{1}$$

For each $R_j = \{(a_{i_1 l_1}, \ldots, a_{i_{k_j} l_{k_j}})\}$ introduce a variable $y_j \geq 0$ and constraints

$$y_j \leq x_{i_q l_q} \tag{2}$$

for $1 \le q \le k_j$. With these constraints, the objective function is

$$\text{maximize } \sum_j w_j y_j \tag{3}$$

We denote by (LP1) the linear programming defined by the above equations.

Theorem 1 *There is a deterministic algorithm for* MAX CSP *that achieves an approximation ratio of* $\min_j |T_j|^{\frac{1}{k}-1}$*, where* $k = \max_j k_j$*. For* $k = 2$*, the ratio achieved is* $\min_{j:T_j=S_p\times S_q}\left(\frac{1}{2\sqrt{s_p}}+\frac{1}{2\sqrt{s_q}}\right)^2$.

Proof: We obtain a randomized algorithm from the above linear program, by assigning to an element a_{il} of S_i the probability

$$z_{il} = \frac{k-1}{ks_i} + \frac{x_{il}}{k}.$$

One can show that

$$z_{il} \ge \left(\frac{x_{il}}{{s_i}^{k-1}}\right)^{\frac{1}{k}}$$

by studying the first derivative of the function

$$f(x) = \frac{\left(\frac{k-1}{ks_i}+\frac{x}{k}\right)^k}{x}.$$

The probability of R_j as above is then

$$\begin{aligned}\prod_{q=1}^{k_j} z_{i_q l_q} &\ge \prod_{q=1}^{k_j}\left(\frac{x_{i_q l_q}}{s_{i_q}^{k-1}}\right)^{\frac{1}{k}} \\ &\ge \frac{{y_j}^{\frac{k_j}{k}}}{\left(\prod_{q=1}^{k_j} s_{i_q}\right)^{1-\frac{1}{k}}} \\ &\ge \frac{y_j}{|T_j|^{1-\frac{1}{k}}}.\end{aligned}$$

Since this bound is a factor of $|T_j|^{\frac{1}{k}-1}$ off from the contribution of y_j to the objective function of the above linear program, it follows that the minimum over j of this quantity is a lower bound on the ratio between the expectation of the total weight of a solution and the linear programming upper bound. This algorithm is in turn derandomized using Lemma 1.

While the preceding proof and bound are very similar to those proved in [12] we present them for completeness so that we can compare this bound with another bound that we present in the next section.

If $k = 2$, then when $k_j = 1$ and $A_j = \{i_1\}$, the probability of R_j is $z_{i_1 l_1} \ge y_j/2$; and when $k_j = 2$ and $A_j = \{i_1, i_2\}$, the probability of R_j is

$$\begin{aligned}z_{i_1 l_1} z_{i_2 l_2} &\ge \left(\frac{1}{2s_{i_1}}+\frac{y_j}{2}\right)\left(\frac{1}{2s_{i_2}}+\frac{y_j}{2}\right) \\ &\ge \left(\frac{1}{2\sqrt{s_{i_1}}}+\frac{1}{2\sqrt{s_{i_2}}}\right)^2 y_j,\end{aligned}$$

where the last inequality holds with equality at $y_j = \frac{1}{2\sqrt{s_{i_1}s_{i_2}}}$. The stated bound and derandomization are then as before. ■

4 Combinatorial Algorithm

In this section we introduce the linear program (LP2) by considering the same objective function (3), and equations (1) with (LP1), but replacing for each variable y_j the inequality constraints (2) with the following equality constraint

$$y_j = (\sum_{q=1}^{k_j} s_{i_q} x_{i_q l_q}) / s_{j\text{tot}} \tag{4}$$

where $s_{j\text{tot}} = \sum_{q=1}^{k_j} s_{i_q}$. The objective function is then

$$\sum_j w_j y_j = \sum_{i,l} r_{il} x_{il}$$

with

$$r_{il} = s_i \sum w_j / s_{j\text{tot}},$$

where this last sum ranges over all the constraints that the variable x_{il} appears in, i.e. it ranges over all $R_j = \{(a_{i_1 l_1}, \ldots, a_{i_{k_j} l_{k_j}})\}$ such that $il = i_q l_q$ for some $1 \le q \le k_j$. Note that the difference between (LP1) and (LP2) is that for each constraint j in (LP1) y_j is the minimum of all the $x_{i_q l_q}$ variables appearing in the constraint while in (LP2) y_j is a convex combination of them.

The objective function above is maximized by selecting for each i the largest r_{il}, setting $x_{il} = 1$, and setting $x_{il'} = 0$ for $l' \neq l$. This observation implies that (LP2) can be solved in a straightforward combinatorial fashion. Furthermore, the following Theorem provides the necessary approximation ratio for the solution of (LP2).

Theorem 2 *There is a combinatorial algorithm for* MAX CSP *that achieves an approximation ratio of* $\min_j |T_j|^{-1} \frac{k-1+s_{j\text{tot}}}{k} (1 + \frac{1}{k-1})^{1-k_j}$, *where* $s_{j\text{tot}} = \sum_{i \in A_j} s_i$ *and* $k = \max_j k_j$. *The running time of the algoroithm is* $O(\sum_{j=1}^{m} |R_j| k_j)$.

Proof: The algorithm takes the values x_{il} obtained by solving (LP2) and again sets

$$z_{il} = \frac{k-1}{k s_i} + \frac{x_{il}}{k}.$$

Writing $u_q = s_{i_q} x_{i_q l_q}$, the probability of R_j is

$$\prod_{q=1}^{k_j} z_{i_q l_q} = \prod_{q=1}^{k_j} \frac{k-1+u_q}{k s_{i_q}}$$

$$\ge |T_j|^{-1} \left(\frac{k-1}{k}\right)^{k_j-1} \left(\frac{k-1+\sum_{q=1}^{k_j} u_q}{k}\right)$$

$$\ge |T_j|^{-1} \left(\frac{k-1}{k}\right)^{k_j-1} \left(\frac{k-1+s_{j\text{tot}}}{k}\right) y_j,$$

where the first inequality follows from the fact that the minimum of $(k-1+u_1)(k-1+u_2)$ subject to $u_1+u_2=c$ and $u_1,u_2\geq 0$ is at $u_1=0$ and $u_2=c$; the second inequality follows from $\sum_{q=1}^{k_j} u_q = s_{j\text{tot}}y_j$. This proves the stated bound, and the derandomization is achieved again with Lemma 1.

The running time for the derandomization is known to be $O(\sum_{j=1}^{m}|R_j|k_j)$. It remains to see that the running time for solving (LP2) is also $O(\sum_{j=1}^{m}|R_j|k_j)$. This requires us to compute $s_{j\text{tot}}$ for each constraint j and r_{il} for all those variables x_{il} that appear in some constraint. The bound $O(\sum_{j=1}^{m}|R_j|k_j)$ follows. ■

Note that the last condition $\sum_{q=1}^{k_j} u_q = \sum_{q=1}^{k_j} s_{i_q}x_{i_ql_q} \geq \sum_{q=1}^{k_j} s_{i_q}y_j = s_{j\text{tot}}y_j$ holds true for (LP1) as well, since y_j is at most the minimum of all the $x_{i_ql_q}$'s. Therefore the analysis above holds for (LP1) as well. We get the corollary:

Corollary 2 *The linear programming algorithm* (LP1) *satisfies a bound of*

$$\min_j \max(|T_j|^{\frac{1}{k}-1}, B_j),$$

where $B_j = |T_j|^{-1}\frac{k-1+s_{j\text{tot}}}{k}(1+\frac{1}{k-1})^{1-k_j}$.

We now analyse the approximation ratio of our simple combinatorial algorithm for the special case when all the variables have the same domain size, namely all s_i are equal to s. Choose some $0<\alpha<1$ and set

$$z_{il} = \frac{1-\alpha}{s} + \alpha x_{il}.$$

Note that we are slightly modifying the algorithm from that analysed earlier where α was set to $1/k$.

Remember that the solution to (LP2) has a very simple structure where for each $i\in[n]$ only one of the x_{il} is set to 1 and the rest $x_{il'}, l'\neq l$ are set to 0. Consider $p\geq 1$ such that p of the $x_{i_ql_q}$ equal 1 and the remaining k_j-p equal 0. The probability of R_j is then

$$\begin{aligned}&\left(\frac{1-\alpha}{s}+\alpha\right)^p\left(\frac{1-\alpha}{s}\right)^{k_j-p}\frac{k_j}{p}y_j\\ &=\left(1+\frac{s\alpha}{1-\alpha}\right)^p\left(\frac{1-\alpha}{s}\right)^{k_j}\frac{k_j}{p}y_j\\ &\geq A_p(\alpha)y_j,\end{aligned}$$

where

$$A_p(\alpha) = \left(1+\frac{s\alpha}{1-\alpha}\right)^p\left(\frac{1-\alpha}{s}\right)^{k}\frac{k}{p}.$$

The algorithm thus achieves a ratio of $A=\max_\alpha \min_p A_p(\alpha)$. We get the following lemma for the above approximation ratio.

Lemma 2 *The approximation ratio achieved by the combinatorial algorithm is* $A = \max_\alpha \min_p A_p(\alpha)$, *where*

$$A_p(\alpha) = \left(1 + \frac{s\alpha}{1-\alpha}\right)^p \left(\frac{1-\alpha}{s}\right)^k \frac{k}{p}$$

Let α^* *be the maximizing* α, *and let* p^* *be the smallest corresponding minimizing* p. *We have* $p^* = \lceil \frac{k-1}{s} \rceil$, *and for* $p = p^*$, $\alpha^* = \frac{1}{sp+1}$ *if* $k \geq \frac{p(sp+1)}{p+1}$, $\alpha^* = \frac{sp-k}{(s-1)k}$ *if* $k \leq \frac{p(sp+1)}{p+1}$. *Therefore the ratio achieved is*

$$A = \left(1 + \frac{1}{p}\right)^p \left(\frac{p}{sp+1}\right)^k \frac{k}{p}$$

if $\frac{p(sp+1)}{p+1} \leq k \leq sp+1$, *and*

$$A = \left(\frac{(s-1)p}{k-p}\right)^{p-1} \left(\frac{k-p}{(s-1)k}\right)^{k-1}$$

if $s(p-1)+1 \leq k \leq \frac{p(sp+1)}{p+1}$.

Proof: The proof proceeds by examining the expression $R_p(\alpha) = \frac{A_p(\alpha)}{A_{p+1}(\alpha)}$, and noting that for a given α, $A_p(\alpha)$ is minimized at p if and only if $R_p \leq 1 \leq R_{p-1}$. Since

$$\frac{A_p(\alpha)}{A_{p+1}(\alpha)} = \left(1 + \frac{1}{p}\right)\left(1 + \frac{s\alpha}{1-\alpha}\right)^{-1}$$

the condition $R_p \leq 1 \leq R_{p-1}$ can be expressed as $p - 1 \leq \frac{1-\alpha}{s\alpha} \leq p$, or equivalently

$$\frac{1}{sp+1} \leq \alpha \leq \frac{1}{s(p-1)+1}$$

This implies that in order to estimate A, it is sufficient for each p to maximize $A_p(\alpha)$ only for values $\alpha \in I_p = [\frac{1}{sp+1}, \frac{1}{s(p-1)+1}]$. Let α_p^* be the value of I_p that maximizes $A_p(\alpha)$. Letting

$$\hat{\alpha}_p = \frac{sp-k}{(s-1)k}$$

we observe that $A_p(\alpha)$ is increasing for $\alpha < \hat{\alpha}_p$ and decreasing for $\alpha > \hat{\alpha}_p$. Thus, α_p^* is equal to $\hat{\alpha}_p$ if $\hat{\alpha}_p$ is a point inside the range I_p; otherwise α_p^* is one of the boundaries of I_p. More specifically

(1) $\alpha_p^* = \hat{\alpha}_p$ if and only if $\frac{1}{sp+1} \leq \hat{\alpha}_p \leq \frac{1}{s(p-1)+1}$ if and only if $s(p-1)+1 \leq k \leq \frac{p(sp+1)}{p+1}$,

(2) $\alpha_p^* = \frac{1}{s(p-1)+1}$ if and only if $\hat{\alpha}_p \geq \frac{1}{s(p-1)+1}$ if and only if $k \leq s(p-1)+1$, and

(3) $\alpha_p^* = \frac{1}{sp+1}$ if and only if $\hat{\alpha}_p \leq \frac{1}{sp+1}$ if and only if $k \geq \frac{p(sp+1)}{p+1}$.

Note that $\alpha^* = \alpha_p^*$ for $p = p^*$. It remains to show that $s(p-1)+1 < k \leq sp+1$ for $p = p^*$. If $k \leq s(p-1)+1$, then $\alpha^* = \frac{1}{s(p-1)+1}$ by (2). Then $A_{p-1}(\alpha^*) = A_p(\alpha^*)$, and we would have $p^* \leq p-1$.

If $k > sp+1$, then $\alpha^* = \frac{1}{sp+1}$ by (3), and so $A_{p+1}(\alpha^*) = A_p(\alpha^*)$. However, $\alpha_{p+1}^* \neq \alpha^*$ by (2) applied to $p+1$, so $A_{p+1}(\alpha_{p+1}^*) > A_{p+1}(\alpha^*)$, contrary to the optimality of α^*. ∎

The case $p^* = \lceil \frac{k-1}{s} \rceil = 1$ gives a ratio $2k(\frac{1}{s+1})^k$ for $k-1 \leq s \leq 2k-1$, and a ratio $(\frac{k-1}{k(s-1)})^{k-1}$ for $s \geq 2k-1$.

When $k=2$, the ratio achieved is thus $\frac{1}{2(s-1)}$ for $s \geq 3$, and $\frac{4}{9}$ for $s=2$. The following result shows that this last ratio cannot be improved with respect to the linear programming upper bound of the combinatorial algorithm.

Theorem 3 *There are instances of* MAX DICUT *where the ratio between the optimal solution and the linear programming upper bound for the combinatorial algorithm is* $\frac{4}{9}(1+O(n^{-\frac{1}{2}}))$.

Proof: MAX DICUT is the problem of partitioning the vertices of a directed graph into two sets S and T so as to maximize the number of edges directed from S to T.

The instances consist of two cliques K and L on n vertices, with edges directed in both directions; and directed edges from vertices in K to vertices in L, with each such directed edge present with probability $\frac{1}{2}$. We shall show that with high probability, the linear programming upper bound is $\frac{3}{2}n^2(1-O(n^{-1}))$, while the optimal solution is $\frac{2}{3}n^2(1+O(n^{-\frac{1}{2}}))$.

For a vertex i, let d_i^{in} and d_i^{out} denote its in-degree and out-degree. The linear programming upper bound is then

$$\sum_i \frac{1}{2}\max(d_i^{\text{in}}, d_i^{\text{out}}) = \sum_{i\in K}\frac{1}{2}d_i^{\text{out}} + \sum_{i\in L}\frac{1}{2}d_i^{\text{in}} = n(n-1) + r_{KL},$$

where r_{KL} is the number of edges directed from K to L. The expected value of r_{KL} is $\frac{1}{2}n^2$, and by the Chernoff bound,

$$Pr(r_{KL} < \frac{1}{2}n^2 - cn) < e^{-c^2}.$$

The linear programming upper bound is thus at least $\frac{3}{2}n^2(1-O(n^{-1}))$ with high probability.

For a given solution, let $A = K \cap S$ and $B = L \cap T$; let $a = |A|$ and $b = |B|$. The value of the solution is then $a(n-a) + b(n-b) + r_{AB}$, where r_{AB} is the number of edges directed from A to B. The expected value of r_{AB} is $\frac{1}{2}ab$, and by the Chernoff bound,

$$Pr(r_{AB} > \frac{1}{2}ab + c\sqrt{abn}) < e^{-c^2 n}.$$

This quantity added over the 2^{2n} choices of A and B is very small for c large enough. Since $\sqrt{abn} = O(n^{\frac{3}{2}})$, the value of the solution is at most $a(n-a)+b(n-b)+\frac{1}{2}ab+O(n^{\frac{3}{2}})$ with high probability. This expression is maximized for $a = b = \frac{2}{3}n$, where it equals $\frac{2}{3}n^2(1+O(n^{-\frac{1}{2}}))$. ∎

Recall Lemma 2. The bounds for the approximation ratio for $p^* \geq 2$ given above are more complicated. We shall instead, for $s \leq k-1$, consider the value $\alpha = \frac{1}{k}$ as before and obtain simpler bounds.

Lemma 3 *With $\alpha = \frac{1}{k}$, the approximation ratio $A_p(\alpha)$ (see Lemma 2) achieved is greater than $s^{1-k}(1 - \frac{s-1}{2(k-1)})$ for $s \leq k-1$.*

Proof: We have

$$A_p(1/k) = \left(1 + \frac{s}{k-1}\right)^p s^{-k} \left(\frac{k-1}{k}\right)^k \left(\frac{k}{p}\right).$$

We set $x = p$ and $a = \frac{s}{k-1}$. The function $f(x) = \frac{1}{x}(1+a)^x$ is minimized at $x = \frac{1}{\ln(1+a)}$, where it equals $e \ln(1+a)$. Therefore

$$A_p(1/k) \geq s^{1-k} \frac{k-1}{s} e \ln\left(1 + \frac{s}{k-1}\right) \left(\frac{k-1}{k}\right)^{k-1}$$

$$\geq s^{1-k} \frac{k-1}{s} \ln(1 + \frac{s}{k-1})(2 - (k-1)\ln(1 + \frac{1}{k-1})) \tag{5}$$

$$\geq s^{1-k} \left(1 - \frac{s-1}{2(k-1)}\right) \tag{6}$$

Inequality (5) asserts that

$$e\left(\frac{k-1}{k}\right)^{k-1} \geq 2 - (k-1)\ln(\frac{k}{k-1})$$

which holds since $ey + \ln(\frac{1}{y}) \geq 2$ for $0 < y < 1$, while inequality (6) follows by first inspecting derivatives with respect to s to show that it is sufficient to consider $s = 2$, and then using power series expansions for the two logarithms. ■

5 Conclusion

We considered the MAX CSP problem over multi-valued domains. We gave an alternate analysis for the linear programming algorithm of Serna et al [12], obtaining a new approximation ratio. Our main contribution was a simple combinatorial algorithm which achieves a ratio within factor e of the linear programming algorithm. For the case of $k = 2$ and $s = 2$, we show an integrality gap of $\frac{4}{9}(1 + O(n^{-\frac{1}{2}}))$ which asymptotically matches our approximation ratio for that case. This proves that our analysis is tight for any method that uses the linear programming upper bound.

References

[1] S. Agrawal, S. Chaudhuri, and V. Narasayya. "Automated Selection of Materialized Views and Indexes for SQL Databases." In *Proceedings of the 26th International Conference on Very Large Databases (VLDB),* 2000, pp. 496–505.

[2] G. Andersson, L. Engebretsen, and J. Håstad. "A new way to use semidefinite programming with applications to linear equations mod p." *Journal of Algorithms,* 2001, 39(2):162-204.

[3] M. Bellare. "Interactive proofs and approximation: reductions from two provers in one round." In *Proceedings of the 2nd Israel Symposium on Theory of Computing Systems,* 1993, pp. 266–274.

[4] S. Chaudhuri, and V. Narasayya. "An efficient, cost-driven index selection tool for Microsoft SQL server." In *Proceedings of the 23rd International Conference on Very Large Databases (VLDB),* 1997, pp. 146–155.

[5] L. Engebretsen. "Lower bounds for non-boolean constraint satisfaction programs." Technical Report TR00-042, Electronic Colloquium on Computational Complexity, June 2000.

[6] U. Feige and M. K. Goemans. "Approximation the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT." In *Proceedings of the 3rd Israel Symposium on Theory of Computing and Systems,* 1995, pp. 182–189.

[7] A. Frieze and M. Jerrum. "Improved approximation algorithms for MAX k-CUT and MAX BISECTION." *Algorithmica,* 1997, 18:67-71. In *Proceedings of the 3rd Israel Symposium on Theory of Computing and Systems,* 1995, pp. 182–189.

[8] M. X. Goemans and D. P. Williamson. ".878-approximation algorithms for MAX CUT and MAX 2SAT." In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing,* 1994, pp. 422–431.

[9] M. X. Goemans and D. P. Williamson. "Approximation algorithms for MAX 3-CUT and other problems via complex semidefinite programming." In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing,* 2001.

[10] E. Halperin and U. Zwick. "Combinatorial approximation algorithms for the maximum directed cut problem" In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms,* 2001, pp. 1-7.

[11] J. Håstad. "Some optimal inapproximability results." In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing,* 1997, pp. 1–10.

[12] M. Serna, L. Trevisan, and F. Xhafa. "The (parallel) approximability of non-boolean satisfiability problems and restricted integer programming." In *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science 1373, Springer-Verlag, 1998, pp. 488–498.