Approximating MAX kCSP using random restrictions

Gustav Hast

June 7, 2004

Abstract

In this paper we study the approximability of the maximization version of constraint satisfaction problems. We provide two probabilistic approximation algorithms for MAX kCONJSAT which is the problem to satisfy as many conjunctions, each of size at most k, as possible. As observed by Trevisan, this leads to approximation algorithms with the same approximation ratio for the more general problem MAX kCSP, where instead of conjunctions arbitrary k-ary constraints are imposed.

The first algorithm achieves an approximation ratio of $2^{1.40-k}$. The second algorithm achieves a slightly better approximation ratio of $2^{1.54-k}$, but the ratio is shown using computational evidence. These ratios should be compared with the previous best algorithm, due to Trevisan, that achieves an approximation ratio of 2^{1-k}. Both the new algorithms use a combination of random restrictions, a method which have been used in circuit complexity, and traditional semidefinite relaxation methods.

A consequence of these new approximation algorithms is that some complexity classes described by probabilistical checkable proofs can be characterized as subsets of P. More specifically we show that $PCP_{c,s}[\log, k] \subseteq P$ if $c/s > 2^{k-1.40}$ and have computational evidence that it is still valid if $c/s > 2^{k-1.54}$.

1 Introduction

In this paper we study the approximability of maximum k-constraint satisfaction problem (MAX kCSP). An instance of MAX kCSP consists of a set of Boolean variables and a set of weighted constraints, each acting over a k-tuple of the variables. The value of an assignment is the sum of the weights of the satisfied constraints. This problem was defined by Khanna et. al. [11] and is a natural generalization of many well-known optimization problems. In fact it can express any MAX SNP problem.

An algorithm r-approximates a maximum optimization problem if the ratio of the value of the solution returned by the algorithm, and the optimal value of the problem, is at least r. Trevisan observed in [14] that it is sufficient to r-approximate a restricted version of MAX kCSP in order to r-approximate MAX kCSP. The restricted version is called MAX kCONJSAT and allows only constraints that are conjunctions of literals. The, to date, best approximation algorithm for MAX kCONJSAT, and thereby also for MAX kCSP, is based on a linear relaxation and is due to Trevisan [14]. It achieves an approximation ratio of 2^{1-k}. Since the work was first published in 1996, it has remained the best algorithm known and one could start to wonder if it, in fact, is the best possible polynomial-time algorithm for MAX kCSP. In this work we provide two algorithms, with approximation ratios of $2^{1.40-k}$ and $2^{1.54-k}$, that show that this is not the case. The ratio of $2^{1.54-k}$ is shown using computational evidence rather than a proof. However, for the approximation ratio of $2^{1.40-k}$ we have a complete proof. The new approximation ratios can be considered as a moderate improvement, but structurally interesting.

In probabilistic proof checking, the problem of maximizing the accepting probability of a verifier looking at k bits can be described as an instance of MAX kCONJSAT [14]. Thus, a consequence of the new approximation ratios is that certain PCP classes can be characterized as subsets of P. For any $c/s > 2^{k-1.40}$ we have that $PCP_{c,s}[\log, k] \subseteq P$, and we have computational evidence that it is still valid if $c/s > 2^{k-1.54}$.

We point out that there exist better algorithms for MAX kCSP if the instances are satisfiable. In that case, another algorithm by Trevisan [15] achieves a ratio of $2^{\log(k+1)-k}$.

In another research direction, PCP techniques have been used to show that approximating MAX kCSP within certain ratios, in polynomial-time, cannot be done unless $P = NP$. A long line of results led to the work of Samorodnitsky and Trevisan [13], which show that there cannot exist a polynomial-time $2^{2\sqrt{k+1}-(k+1)}$-approximation algorithm for MAX kCSP unless $P = NP$. More recently Engebretsen and Holmerin [2] achieved an even stronger result showing that, for $k \geq 3$, there cannot exist a polynomial-time $2^{\sqrt{2k-2}+1/2-k}$-approximation algorithm.

Semidefinite relaxation. Let us give a brief introduction into semidefinite relaxations in connection with approximation algorithms. The work of Goemans and Williamson [5] laid ground for a large number of works providing better approximation ratios for various constraint satisfaction problems. The idea is to relax each clause into a set of semidefinite constraints. The resulting semidefinite program can be solved to any desired accuracy in polynomial-time. The solution is then rounded into a solution of the original constraint satisfaction problem. In some cases it is possible to analyze the rounding technique used and thereby obtain an approximation ratio. For MAX 2CONJSAT Goemans and Williamson [5] obtained a 0.79607-approximation algorithm. Later, Karloff and Zwick [10] introduced a methodology to create semidefinite relaxations for any constraint function and tools to analyze the rounding scheme in some cases. Using these methods Zwick [17] obtained a 0.5-approximation algorithm for MAX 3CONJSAT and Guruswami et. al. [7] obtained a 0.33-approximation algorithm for MAX 4CONJSAT. We note that these algorithms outperform the random assignment algorithm with more than a factor of 3, 4 and 5, respectively. In contrast, the general MAX kCONJSAT by Trevisan, which is based on a linear relaxation instead of a semidefinite relaxation, beats the random assignment algorithm with a factor of 2. Unfortunately, applying the semidefinite relaxation method on MAX kCONJSAT for

$k > 4$ makes the analysis of the approximation ratio problematic. An interesting question that remains open is if the algorithm achieves an approximation ratio of $2^{\log k - k}$, as it does for $k \leq 4$, but we simply do not know how to show it.

Techniques used. The main idea of both our MAX kCONJSAT algorithms is to make use of the fact that semidefinite relaxation methods work well for conjunctions of small size. We shrink the size of the conjunctions by using random restrictions, an often used method in circuit complexity that was introduced by Furst et. al. [4]. The random restriction is done by assigning a random value to each variable with high probability. Many of the conjunctions are unsatisfiable after this random restriction and are thrown away, but the effective sizes of the remaining conjunctions have decreased so they are tractable for traditional semidefinite relaxation methods. For the $2^{1.54-k}$ algorithm we use the above mentioned methodology of Karloff and Zwick and analyze its effect on the restricted instance. For the $2^{1.40-k}$ algorithm we instead use so-called gadgets [16] to transform the restricted instance of conjunctions into a MAX 2SAT instance which is solved using the algorithm of Feige and Goemans [3].

When analyzing the approximation ratio of our $2^{1.54-k}$ algorithm we have to rely on the assumption that we have found global optima of two continuous optimization problems, one in ten variables and one in six variables. We note that this situation is quite common for many works that analyze the effectiveness of rounding procedures of semidefinite programs [3, 7, 8, 10, 17].[1] In the case of our $2^{1.40-k}$ algorithm we rely on the Feige-Goemans algorithm which has been rigorously analyzed by Zwick [18].

Probabilistic vs. deterministic algorithms. We note that both our algorithms are probabilistic and thus the approximation ratio is calculated using the expected value of the solution returned by the algorithm. We do not see any problem derandomizing them with the method of Mahajan and Ramesh [12], but we have not studied it closely. Alternatively, multiple independent executions of the algorithm guarantees that except with exponentially small probability a solution with value arbitrarily close to the one stipulated is produced. For some applications only a lower bound of the optimum value is needed and we note that both algorithms can be used to get such a bound in a deterministic way (see Section 6).

2 Preliminaries

We say that an instance is ρ-satisfiable if the ratio of an optimal solution and the total weight of all clauses is at least ρ. An algorithm A α-approximates a maximization problem if for all instances x of the problem $\mathrm{val}(A, x)/\mathrm{opt}(x) \geq \alpha$, where $\mathrm{val}(A, x)$ is the value of $A(x)$ and $\mathrm{opt}(x)$ is the optimal value of x. Equivalently, A is said to have an approximation

[1] Some of these results have been proved in a rigorous manner since the original publication and many of the assumptions are seemingly not as strong as our is, i.e. involve less variables.

ratio of α. For probabilistic algorithms $\text{val}(A, x)$ is allowed to be an expected value over the random choices done by A.

An instance of the Max kCSP problem consists of a set $\{C_1, \ldots, C_m\}$ of clauses (or constraints) with associated weights $\{w_1, \ldots, w_m\}$ and a set of Boolean variables $X = \{x_1, \ldots, x_n\}$. Each clause C_i consists of a Boolean function f_i of arity k and a size k tuple of Boolean variables $(x_{i_1}, \ldots, x_{i_k})$ where $x_{i_j} \in X$. A solution is an assignment to X and the value of the solution is the sum of the weights of the satisfied clauses. A clause $C_i = (f_i, (x_{i_1}, \ldots, x_{i_k}))$ is satisfied if $f_i(x_{i_1}, \ldots, x_{i_k})$ is true.

An instance of the Max kConjSAT problem is a special type of Max kCSP problem where each clause is a conjunction of literals from X. A literal is either a variable or a negated variable.

3 A $2^{1.40-k}$-approximation algorithm for Max kConjSAT

We begin this section by giving a description of our first Max kConjSAT algorithm and after that give a detailed analysis of the approximation ratio. The algorithm runs both a random assignment algorithm and Algorithm $A1$, depicted in Figure 1, and returns the best solution.

The random assignment algorithm assigns a random Boolean value to each variable in the instance. The expected value of the solution is at least a 2^{-k}-fraction of the total weight of all clauses. Thus, if the instance is not $2^{-1.40}$-satisfiable then the random assignment algorithm is a $2^{1.40-k}$-approximation.

If the instance instead is $2^{-1.40}$-satisfiable then we show that Algorithm $A1$ produces a $2^{1.40-k}$-approximation. The algorithm starts by applying a random restriction over the instance, thereby reducing the size of the conjunctions. The small conjunctions are then transformed, using gadgets, into a Max 2SAT instance which is solved using the algorithm of Feige and Goemans [3].

In this section we prove the following theorem.

Theorem 1. Max kConjSAT *has a polynomial-time* $2^{1.40-k}$*-approximation algorithm.*

Using the observation made by Trevisan [14], the above theorem yields the corresponding result for Max kCSP.

Corollary 2. Max kCSP *has a polynomial-time* $2^{1.40-k}$*-approximation algorithm.*

First we describe two techniques used in the algorithm; Section 3.1 explains random restrictions and Section 3.2 explains gadgets. After this we analyze the algorithm and prove Theorem 1.

Input: A set of Boolean variables $\{x_1, \dots x_n\}$ and a set of conjunctions $\{C_1, \dots C_m\}$ and weights $\{w_1, \dots w_m\}$.

1. **(Initialization)** Set the parameter $s := 2.56$.
2. **(Random restriction)** Set $I := \emptyset$. For each $i := 1, \dots n$ do:
   - With probability s/k: put i in I, $I := I \cup \{i\}$.
   - With remaining probability: let x_i be assigned according to an unbiased coin flip.
3. **(Gadget)** Set $\mathcal{C}_{2\text{SAT}} := \emptyset$. For each $j := 1, \dots m$ do: If C_j still is satisfiable given the already set variables and include at most five unassigned variables, apply the gadget to 2SAT as explained in Section 3.2. Put the 2SAT clauses in $\mathcal{C}_{2\text{SAT}}$, each with weight w_j.
4. **(Solve** Max 2SAT**)** Apply the Feige-Goemans algorithm [3] on $\mathcal{C}_{2\text{SAT}}$ and let it assign the remaining variables $\{x_i\}_{i \in I}$.

Figure 1: Algorithm $A1$: a Max kConjSAT algorithm.

3.1 Random restrictions

In the introduction it was mentioned that semidefinite relaxation methods works well on small conjunctions. A Max kConjSAT instance consists of length-k conjunctions and in order to reduce the length of these conjunctions, and thereby making the instance tractable for semidefinite methods, we use a technique called random restrictions. For each variable, either assign it a random value with probability $1 - s/k$, or let it be unassigned with probability s/k. Each clause is originally a conjunction of at most k variables. Restricting a clause to the values set by the random restriction either makes the clause true (if all literals in the clause were made true by the restriction), false (if any of the literals in the clause were made false by the restriction) or it is still a conjunction but with size i (if all but i literals were set and made true by the restriction).

3.2 Gadgets reducing ConjSAT clauses to 2SAT clauses

A gadget is a way to transform a constraint of one type into a set of constraints of another type. In [16] Trevisan et. al. exhibit a gadget reducing a conjunction of three literals, $X_1 \wedge X_2 \wedge X_3$ into 2SAT clauses. A 2SAT clause is a disjunction of literals of size at most 2. The gadget is Y, $(\neg Y \vee X_1)$, $(\neg Y \vee X_2)$, $(\neg Y \vee X_3)$, where Y is a so-called auxiliary variable that is added. Note that if $X_1 \wedge X_2 \wedge X_3$ is true, then all four of the gadget's clauses can be made true by setting Y to true. However, if $X_1 \wedge X_2 \wedge X_3$ is false then we can never satisfy all four clauses, but always three of them by setting Y to false. For those familiar with the jargon of [16], this is a strict 4-gadget. In [16] this gadget was used to create a 0.367-approximation algorithm for MAX 3CONJSAT.

Generalizing this gadget construction we can produce a gadget reducing a conjunction of size i, $X_1 \wedge \ldots \wedge X_i$ into $i+1$ 2SAT clauses: Y, $(\neg Y \vee X_1)$, $\ldots (\neg Y \vee X_i)$. (For $i = 1$ we do not have to use a gadget because X_1 is already a 2SAT clause.) Using the above gadget it is easy to generalize the MAX 3CONJSAT algorithm by Trevisan et. al., achieving approximation ratios of $2^{1.52-4}$ for MAX 4CONJSAT, $2^{1.37-5}$ for MAX 5CONJSAT and $2^{1.18-6}$ for MAX 6CONJSAT. For $k > 6$, MAX kCONJSAT is approximated better by the 2^{1-k}-approximation algorithm of Trevisan.

3.3 Analysis of the approximation ratio

As already mentioned the random assignment algorithm produce $2^{1.40-k}$-approximations on instances that are not $2^{-1.40}$-satisfiable. Thus, left to be shown is that Algorithm $A1$ produce $2^{1.40-k}$-approximations on instances that are $2^{-1.40}$-satisfiable. We can assume that each conjunction in the instance is of length k, because such a conjunction can be padded to length k with auxiliary variables that simultaneously can be satisfied without effecting the value of a solution. In fact, a length-k conjunction is the worst case for the analysis of the approximation ratio and thus there is no need for padding in the algorithm.

The first phase of Algorithm $A1$ is to apply a random restriction, with $s = 2.56$ (see Section 3.1). This yields an instance of fairly small conjunctions. On each such iCONJSAT clause containing at most five literals we apply the gadget mentioned above, which reduces the clause to i length-2 and one length-1 2SAT clauses. If $i = 1$, we do not have to use a gadget at all because a single literal is also a 2SAT clause. Thus we get a MAX 2SAT instance that can be solved using the MAX 2SAT algorithm of Feige and Goemans [3] thereby deciding the variables that remained unassigned after the random restriction.

We only use conjunctions with at most five literals because the gadget construction is less effective for larger conjunctions than for smaller ones. Assume that a ρ-fraction of the length-i conjunctions are satisfiable and β is the approximation ratio of the MAX 2SAT algorithm. The analysis gives that the expected fraction of satisfied conjunctions is at least $\beta(i+\rho)-i$ and thus for large enough conjunctions there will be a negative effect to include them.

The probability that an arbitrary conjunction of length k remains satisfiable after the random restriction and contains exactly i unassigned variables is

$$p_i = \binom{k}{i} \left(\frac{s}{k}\right)^i \left(1 - \frac{s}{k}\right)^{k-i} 2^{-(k-i)}. \tag{1}$$

Let w_{tot} be the sum of all weights in the MAX kCONJSAT instance. The expected sum of weights from length-1 clauses that an instance induce is $\alpha_1 w_{\mathrm{tot}}$ and the corresponding value for length-2 clauses is $\alpha_2 w_{\mathrm{tot}}$, where $\alpha_1 = \sum_{i=1}^{5} p_i$ and $\alpha_2 = \sum_{i=2}^{5} i p_i$ are the expected number of 2SAT clauses, length-1 respectively length-2, induced from a single conjunction of length k.

We can assume that the original instance is $2^{-1.40}$-satisfiable. Let the value of an optimal assignment be $w_{\text{opt}} = \rho w_{\text{tot}}$, where $\rho \geq 2^{-1.40}$. Such an assignment, restricted to the unassigned variables, satisfy all of the length-2 clauses and an expected ρ-fraction of the weights of the length-1 clauses in the Max 2SAT instance. Thus, the expected weight of satisfied length-2 clauses is $\alpha_2 w_{\text{tot}}$ and the expected weight of satisfied length-1 clauses is $\rho\alpha_1 w_{\text{tot}}$.

Using the Max 2SAT algorithm of Feige and Goemans [3] we are guaranteed to obtain a solution of expected weight at least $\beta_1\rho\alpha_1 w_{\text{tot}} + \beta_2\alpha_2 w_{\text{tot}}$, where $\beta_1 = 0.976$ and $\beta_2 = 0.931$. The use of gadgets ensures that if we let the solution decide the value of the unassigned variables in the original Max kConjSAT instance, the weight of that assignment will have expected value at least $\beta_1\rho\alpha_1 w_{\text{tot}} + \beta_2\alpha_2 w_{\text{tot}} - \alpha_2 w_{\text{tot}} + p_0 w_{\text{tot}}$. The contribution of $p_0 w_{\text{tot}}$ is due to the event that all variables in a clause were already set by the random assignment and it made the clause true. We can now express an approximation ratio:

$$\frac{\mathrm{E}\left[w_{\text{app}}\right]}{w_{\text{opt}}} \geq \frac{\beta_1\rho\alpha_1 w_{\text{tot}} + \beta_2\alpha_2 w_{\text{tot}} - \alpha_2 w_{\text{tot}} + p_0 w_{\text{tot}}}{\rho w_{\text{tot}}} = \beta_1\alpha_1 - \rho^{-1}(1-\beta_2)\alpha_2 + \rho^{-1}p_0.$$

We study the asymptotic behavior of p_i in order to calculate the approximation ratio when $k \to \infty$.

$$p_i = \binom{k}{i}\left(\frac{s}{k}\right)^i\left(1-\frac{s}{k}\right)^{k-i}2^{-(k-i)} \approx \frac{k^i}{i!}\frac{s^i}{k^i}e^{-s}2^{i-k} = \frac{(2s)^i}{e^s i!}2^{-k}$$

Using $s = 2.56$ we get $\alpha_1 = 7.618 \cdot 2^{-k}$, $\alpha_2 = 27.401 \cdot 2^{-k}$ and $p_0 = 0.0773 \cdot 2^{-k}$ thus yielding an approximation ratio greater than $2^{1.40-k}$ for values of k large enough. But we have to show that the ratio is valid for all values of $k \geq 5$. We have done this by numerically checking that the ratio is higher than $2^{1.40-k}$ for all values of k from 5 up to 1000 and then obtaining a lower bound of $2^{1.40-k}$ for all $k \geq 1000$. Details about this can be found in Appendix A.

Instead of using Feige and Goemans Max 2SAT algorithm we could use the one specified by Halperin and Zwick [8, Section 5.4]. That algorithm approximates the number of satisfied length-1 clauses better, $\beta_1 = 0.9828$, but the number of satisfied length-2 clauses slightly worse, $\beta_2 = 0.9309$, compared with the algorithm of Feige and Goemans. Using the algorithm of Halperin and Zwick yields an approximation ratio of $2^{1.41-k}$. However, that algorithm has not been rigorously analyzed as was done by Zwick [18] with the Feige-Goemans algorithm.

4 A $2^{1.54-k}$-approximation algorithm for Max kConjSAT

In this section we present another approximation algorithm for Max kConjSAT. This algorithm also runs two different algorithms: a linear relaxation algorithm, described in

Section 4.1, and Algorithm $A2$, depicted in Figure 2. The linear relaxation algorithm performs well on instances that are $(1+2^{1-k})/2$-satisfiable and algorithm $A2$ achieves well on all other instances.

Algorithm $A2$ is similar in approach to Algorithm $A1$. The first step is to solve a semidefinite relaxation of the problem instance. As in the previous algorithm a random restriction is imposed on the variables (see Section 3.1), but instead of reducing the conjunctions of the unassigned variables into a Max 2SAT instance we use the solution obtained from the semidefinite relaxation to assign values to the remaining variables by performing a random hyperplane rounding.

As in many previous works dealing with approximation algorithms for Max CSP problems using semidefinite programming we only present computational evidence for the approximation ratio. The result is valid if we have found the worst possible vector configuration in two different cases. More explicitly we rely on the following technical assumption.

Assumption 1. *Let $P = 0.72$, $(v_0, v_1, \ldots v_3)$ be a vector configuration of four vectors in R^4 and z be the minimum of rel_A, where $A \subseteq \{1,2,3\}$, defined in Figure 3. Then* 0.493 *is a lower bound on*

$$(P \cdot \Pr_{n_H \in R^4}[sgn(n_h \cdot v_0) = sgn(n_h \cdot v_1) = \ldots sgn(n_h \cdot v_3)] + (1-P)\cdot 2^{-3})/z,$$

for all possible vector configurations of $(v_0, v_1, \ldots v_3)$.

Furthermore, let $(v_0, v_1, \ldots v_4)$ be a vector configuration of five vectors in R^5, and z be the minimum of rel_A, where $A \subseteq \{1,2,3,4\}$, defined in Figure 3. Then 0.337 *is a lower bound on*

$$(P \cdot \Pr_{n_H \in R^5}[sgn(n_h \cdot v_0) = sgn(n_h \cdot v_1) = \ldots sgn(n_h \cdot v_4)] + (1-P)\cdot 2^{-4})/z,$$

for all possible vector configurations of $(v_0, v_1, \ldots v_4)$.

We have computational evidence that vector configurations of four vectors where all pairs of vectors have angle 1.438 are global minima and attain values close to the lower bound of 0.493. Vector configurations of five vectors are defined, up to rotation, by ten pair-wise angles. Let θ_{ij} be the angle between v_i and v_j. If for some a, $\theta_{ai} = 1.855$ for all $i \neq a$ and $\theta_{ij} = 1.424$ for all other pair-wise angles, then the vector configuration seems to be a global minimum and attains a value close to the lower bound of 0.337.

In this section we will show the following theorem. The corollary follows in the same way as in the previous section.

Theorem 3. *If Assumption 1 is true, then* Max kConjSAT *has a polynomial-time $2^{1.54-k}$-approximation algorithm.*

Corollary 4. *If Assumption 1 is true, then* Max kCSP *has a polynomial-time $2^{1.54-k}$-approximation algorithm.*

Input: A set of Boolean variables $\{x_1, \ldots x_n\}$ and a set of conjunctions $\{C_1, \ldots C_m\}$ and weights $\{w_1, \ldots w_m\}$.

1. **(Initialization)** Set the parameters $s := 2.8$, $P := 0.72$.
2. **(Semidefinite programming)** Generate the semidefinite programming relaxation of the input instance given in Figure 3. Find an (almost) optimal solution $v_0, \ldots v_n$ in polynomial-time using standard techniques.
3. **(Random restriction)** Set $I := \emptyset$. For each $i := 1,\ldots n$ do:
   - With probability s/k: put i in I, $I := I \cup \{i\}$.
   - With remaining probability: let x_i be assigned according to an unbiased coin flip.
4. **(Hyperplane rounding / Random assignment)** Choose a random hyperplane normal $n_H \in S^n$. With probability P do for each $i \in I$:
   - If $sgn(n_H \cdot v_0) = sgn(n_H \cdot v_i)$ assign $x_i := 1$.
   - If $sgn(n_H \cdot v_0) \neq sgn(n_H \cdot v_i)$ assign $x_i := 0$.

   With remaining probability let x_i be assigned according to an unbiased coin flip, for all $i \in I$.

Figure 2: Algorithm $A2$: a MAX kCONJSAT algorithm.

The reduction is efficient, thus if the approximation algorithm on an instance does not achieve the claimed approximation ratio, we will be able to construct a vector configuration contradicting Assumption 1.

Next we describe the linear relaxation algorithm and show Theorem 5 that states the performance of the algorithm. After that we describe the semidefinite relaxation and the rounding method used. Finally we analyze the approximation ratio and prove Theorem 3.

4.1 A linear relaxation algorithm for MAX kCONJSAT with threshold rounding

The linear program relaxation is identical to the one by Trevisan [14]. Each binary variable x_i is relaxed to t_i such that $0 \leq t_i \leq 1$. The clause C_j has an associated variable z_j in the relaxation. If x_i occurs positively in C_j, then $z_j \leq t_i$ is a constraint in the relaxation, and if x_i occurs negated then $z_j \leq 1 - t_i$. It is easy to see that an assignment to MAX kCONJSAT can be transformed into a valid assignment for the linear program with the same objective value. This is done by letting $t_i = x_i$.

The difference of this algorithm compared with [14] is in how an assignment to the linear relaxed program is transformed into an assignment of the associated MAX kCONJSAT instance. In [14] the value of x_i was decided by a flip of an unbiased coin with probability $(k-1)/k$ and only in the remaining case had the value of t_i some impact on the value of x_i. We instead make a threshold rounding by setting $x_i = 1$ if $t_i > 1/2$ and $x_i = 0$ otherwise. By analyzing this rounding technique we get the following theorem.

Theorem 5. *A $(1+\epsilon)/2$-satisfiable instance of* MAX kCONJSAT*, where $\epsilon > 0$, can be approximated in polynomial time within $2\epsilon/(1+\epsilon)$ of the optimal value.*

Proof. Let w_{tot} be the sum of all the weights of the clauses, $\sum_{j=1}^{m} w_j$. Furthermore let ϵ' be defined such that $w_{\text{tot}}(1+\epsilon')/2$ is the value of an optimal solution to the MAX kCONJSAT instance. The value of an optimal solution to the linear program, $\sum_{j=1}^{m} w_j z_j^*$, will be at least as large as $w_{\text{tot}}(1+\epsilon')/2$ because it is a relaxation of the original problem. Let $A = \{j : z_j^* \leq 1/2\}$ and $B = \{j : z_j^* > 1/2\}$. Furthermore we know that $z_j \leq 1$. We provide an upper bound of the optimal value.

$$\begin{aligned}
\frac{w_{\text{tot}}(1+\epsilon')}{2} &\leq \sum_{j=1}^{m} w_j z_j^* \\
&= \sum_{j\in A} w_j z_j^* + \sum_{j\in B} w_j z_j^* \\
&\leq \frac{1}{2}\sum_{j\in A} w_j + \sum_{j\in B} w_j \\
&= \frac{1}{2}\sum_{j=1}^{m} w_j + \frac{1}{2}\sum_{j\in B} w_j
\end{aligned}$$

Subtracting $w_{\text{tot}}/2$ from the left and the right equation and multiplying with two we get $\sum_{j\in B} w_j \geq \epsilon' w_{\text{tot}}$. The threshold rounding satisfy all the clauses that have $z_j^* > 1/2$, thus the value of the solution from the threshold rounding is at least $\sum_{j\in B} w_j$. The approximation ratio is thus at least

$$\frac{\sum_{j\in B} w_j}{w_{\text{tot}}(1+\epsilon')/2} \geq \frac{2\epsilon' w_{\text{tot}}}{w_{\text{tot}}(1+\epsilon')} = \frac{2\epsilon'}{1+\epsilon'} \geq \frac{2\epsilon}{1+\epsilon}$$

where we in the last inequality used that $\epsilon' \geq \epsilon > 0$. □

4.2 Semidefinite relaxations

Remember that a MAX CONJSAT instance consists of a set of conjunctions $\{C_1, \ldots, C_m\}$ with weights $\{w_1, \ldots, w_m\}$ and a set of variables $\{x_1, \ldots, x_n\}$. In the semidefinite relaxation each variable x_i corresponds to a vector v_i on the unit sphere S^n in R^{n+1}. A variable can appear negated in a conjunction and thus there is for each x_i a vector $v_{i+n} = -v_i$ which

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{m} w_j z_j \\
z_j \ \leq \ & rel_A,\ A \subseteq C_j, |A| \leq 4 \\
rel_\emptyset \ = \ & 1 \\
rel_{\{i\}} \ = \ & \frac{1 + v_0 \cdot v_i}{2} \\
rel_{\{i,j\}} \ = \ & \frac{1 + v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j}{4} \\
rel_{\{i,j,k\}} \ = \ & \min_{\substack{(\sigma_0,\sigma_1,\sigma_2,\sigma_3)\in \\ permute(\{0,i,j,k\})}} \frac{1 + v_{\sigma_0} \cdot v_{\sigma_1} + v_{\sigma_0} \cdot v_{\sigma_2} + v_{\sigma_1} \cdot v_{\sigma_2}}{4} \\
rel_{\{i,j,k,l\}} \ = \ & \min\Big(\frac{2 + \sum_{\sigma_0,\sigma_1 \in \{0,i,j,k,l\}, \sigma_0 \neq \sigma_1} v_{\sigma_0} \cdot v_{\sigma_1}}{12}, \\
& \min_{\substack{(\sigma_0,\sigma_1,\sigma_2,\sigma_3,\sigma_4)\in \\ permute(\{0,i,j,k,l\})}} \frac{2 - \sum_{t=1}^{4} v_{\sigma_0} \cdot v_{\sigma_t} + \sum_{1 \leq t < u \leq 4} v_{\sigma_t} \cdot v_{\sigma_u}}{4}\Big) \\
v_i \ \in \ & S^n,\ 0 \leq i \leq 2n \\
v_{i+n} \ = \ & -v_i,\ 1 \leq i \leq n
\end{aligned}
$$

Figure 3: Semidefinite program relaxation for MAX kCONJSAT

correspond to $\overline{x_i}$. This is done only for notational convenience and we could replace v_{i+n} by $-v_i$ everywhere. By somewhat abusing previous notation, a conjunction C_j can be described as a subset of $[2n]$ such that if x_i is part of the conjunction then $i \in C_j$ and if $\overline{x_i}$ is part of the conjunction then $i + n \in C_j$. There is also a vector v_0 which represents true. Karloff and Zwick [10] introduced the so-called canonical semidefinite relaxation which specifies how to create the strongest possible relaxations from a specific constraint function. This construction could be applied on conjunctions of k variables, but we only use the canonical semidefinite relaxation of each subset of literals of size at most four. In Figure 3 the complete semidefinite program we use is specified.

Hyperplane rounding is a popular method of obtaining a solution of the original problem from the solution of the semidefinite relaxation and is done as follows. Choose a random hyperplane in R^{n+1}. Set $x_i = 1$ if v_i lies on the same side of the hyperplane as v_0 does, and otherwise set $x_i = 0$. We employ the hyperplane rounding as well, but first after making a random restriction. Thus, only the variables that remain unassigned after the random restriction get their value set by the hyperplane rounding method, all others are set by unbiased coin flips.

4.3 Analysis of the approximation ratio

The linear relaxation algorithm will, according to Theorem 5, approximate any $(1+2^{1-k})/2$-satisfiable instance within at least a factor of $2^{2-k}/(1+2^{1-k})$. This means that for $k \geq 5$ the algorithm at least $2^{1.91-k}$-approximates such an instance. From now on we can therefore assume that the instance is not $(1+2^{1-k})/2$-satisfiable.

Algorithm $A2$ starts by solving the semidefinite program relaxation to (almost) optimality in polynomial-time (e.g. by using the ellipsoid method [6]). For each clause C_j, let z_j^* denote the relaxed value of the clause in the obtained solution. Assume that the clause C_j contains exactly i unassigned variables after the random restriction. Let $q_i(z_j^*)$ denote a lower bound on the probability that these i variables are set according to the conjunction C_j. A lower bound on the expected value of the solution from Algorithm $A2$ can then be expressed as

$$\sum_{j=1}^{m} w_j \sum_{i=0}^{k} p_i q_i(z_j^*),$$

where p_i is given by Equation (1) and is the probability that an arbitrary conjunction of length k remains satisfiable under the random restriction and contains exactly i unassigned variables.

We show the following values of the lower bounds.

$$\begin{aligned}
q_0 &= 1 \\
q_1(z_j^*) &= P \cdot 0.87857 \cdot z_j^* + (1-P) \cdot 2^{-1} \\
q_2(z_j^*) &= P \cdot 0.79607 \cdot z_j^* + (1-P) \cdot 2^{-2} \\
q_3(z_j^*) &= \beta_3(P) \cdot z_j^* \\
q_4(z_j^*) &= \beta_4(P) \cdot z_j^* \\
q_i &= (1-P) \cdot 2^{-i}, \ i \geq 5
\end{aligned}$$

Remember that with probability P the variables are assigned using hyperplane rounding, and with probability $1-P$ a random assignment is used. Algorithm $A2$ specifies that $P = 0.72$ and for that value of P we have that $\beta_3(P) = 0.493$ and $\beta_4(P) = 0.337$ according to Assumption 1.

In the first case there are no unassigned variables in C_j that can make C_j false, thus $q_0 = 1$. If only one variable of the conjunction C_j is assigned by hyperplane rounding, we know that $z_j^* \leq (1 + v_0 \cdot v_l)/2$ according to the semidefinite relaxation, where v_l is assumed to represent the variable that is rounded by the hyperplane. The probability that v_0 and v_l are on the same side of a randomly chosen hyperplane (and thus making the literal corresponding to v_l true) is $1 - \arccos(v_0 \cdot v_l)/\pi$. By substituting $v_0 \cdot v_l$ for $2z_j^* - 1$ calculations give that $1 - \arccos(v_0 \cdot v_l)/\pi \geq 0.87857 z_j^*$. This is essentially the Max Cut analysis of Goemans and Williamson [5]. If the literal instead was assigned a random value, this value is in accordance with C_j with probability 2^{-1}. Thus $q_1(z_j^*)$ is equal to $P \cdot 0.87857 \cdot z_j^* + (1-P) \cdot 2^{-1}$.

The probability that two variables, that were assigned values using hyperplane rounding, are in accordance with the constraint is at least $0.79607 z_j^*$. This is the Max Di-Cut analysis of Goemans and Williamson [5]. Considering the random assignment as well we get $q_2(z_j^*) = P \cdot 0.79607 \cdot z_j^* + (1-P) \cdot 2^{-2}$.

For the cases when three or four variables were rounded using the hyperplane method we turn to the works of Zwick [17] and Guruswami et. al. [7]. In both these cases the ratio obtained assumes that the algorithm is combined with a random assignment with a certain probability P. The approximation ratio β depends upon this probability P. As already mentioned, for $P = 0.72$ we have that $q_3(z_j^*) = 0.493 \cdot z_j^*$ and $q_4(z_j^*) = 0.337 \cdot z_j^*$ as a consequence of Assumption 1.

We have computational evidence for Assumption 1, obtained using the methodology of Karloff and Zwick [10] as explained in [8, Section 6.1]. The minimization was done by the `minimax` function from Matlab's optimization toolbox. Evaluating the probability that hyperplane rounding of a vector configuration satisfies a clause was done by numerical integration of the volume of a spherical tetrahedron using Hsiang's method [9].

If more than four variables in a clause were hyperplane rounded we are unable to analyze the probability that the assignment will satisfy the clause. Thus we only count the success probability for the random assignment in this case.

For fixed values on s and k the values of p_i are just numbers. Furthermore, the lower bounds, $q_i(z_j^*)$, are linear in z_j^*, $q_i(z_j^*) = a_i + b_i z_j^*$. Thus, a lower bound of the expected value of the solution can be expressed on the form

$$\sum_{j=1}^{m} w_j \sum_{i=0}^{k} p_i q_{i,j} = \sum_{j=1}^{m} w_j \sum_{i=0}^{k} p_i \left(a_i + b_i z_j^*\right) = \sum_{j=1}^{m} w_j \left(a + b z_j^*\right) = a \sum_{j=1}^{m} w_j + b \sum_{j=1}^{m} w_j z_j^*, \quad (2)$$

for constants $a = \sum_{i=0}^{k} p_i a_i$ and $b = \sum_{i=0}^{k} p_i b_i$. The value of the obtained solution from the semidefinite program, $\sum_{j=1}^{m} w_j z_j^*$, is known to be at least within a factor $(1-\epsilon)$ of the optimal value of the Max kConjSAT instance, where ϵ is an arbitrarily small positive value.

We study the asymptotic behavior when $k \to \infty$ by making the same approximation of the values of p_i as was done in the analysis of Algorithm $A1$ obtaining $a \geq 0.217 \cdot 2^{-k}$ and $b \geq 2.479 \cdot 2^{-k}$ when $s = 2.8$. The expected approximation ratio can be bounded from below:

$$\begin{aligned} \frac{\mathrm{E}\left[w_{\mathrm{app}}\right]}{w_{\mathrm{opt}}} &\geq \frac{a \sum_{j=1}^{m} w_j + b \sum_{j=1}^{m} w_j z_j^*}{w_{\mathrm{opt}}} \\ &\geq \frac{a}{(1+2^{1-k})/2} + \frac{b \cdot (1-\epsilon) w_{\mathrm{opt}}}{w_{\mathrm{opt}}} \\ &\approx 2a + b, \end{aligned}$$

where we used the fact that the instance not is $(1+2^{1-k})/2$-satisfiable. Algorithm $A2$ achieves an approximation ratio of $2a+b \geq 2^{1.54-k}$ for instance that not are $(1+2^{1-k})/2$-satisfiable, and the linear relaxation algorithm achieves a better ratio for all other instances, thus the combined algorithm $2^{1.54-k}$-approximates Max kConjSAT. The algorithm runs in polynomial-time and thus Theorem 3 is proven.

4.4 An alternative to Algorithm $A2$

From a practical standpoint Algorithm $A2$ seems a bit odd. First it solves a huge semidefinite program and after this only a small fraction of the variables are assigned values according to the solution. A more efficient order seems to be to do the random restriction first, as in Algorithm $A1$, and build a semidefinite program from the small fraction of conjunctions that still are satisfiable. This approach is in fact also realizable, but each weight in the semidefinite program have to be adjusted according to the length of its conjunction.

Remember that the probability that a clause with i literals will be true after performing a combined random hyperplane rounding (with probability P) and random rounding (with probability $1-P$) is at least $q_i(z_j^*)$. Let b_i represent the z_j^* coefficient of the expression for $q_i(z_j^*)$, thus the only non-zero values of b_i are $b_1 = P \cdot 0.87857$, $b_2 = P \cdot 0.79607$, $b_3 = \beta_3(P)$ and $b_4 = \beta_4(P)$.

After applying the random restriction, let $\mathcal{C}_i$ be the set of indices of the conjunctions that still are satisfiable and contain exactly i unassigned variables. Make a semidefinite relaxation of each conjunction C_j, $j \in \cup_{i=0}^{k} \mathcal{C}_i$, as stipulated by Figure 3, but where the weight of a conjunction C_j, $j \in \mathcal{C}_i$, is $b_i w_j$ instead of w_j. Obtain an (almost) optimal solution to the relaxed problem and make a hyperplane rounding with probability P and apply a random assignment with probability $1-P$, where still $P = 0.72$. We now express a lower bound of the expected value of the solution.

$$\mathrm{E}\left[\sum_{i=0}^{k}\sum_{j\in\mathcal{C}_i} w_j q_i(z_j^*)\right] = \mathrm{E}\left[\sum_{i=0}^{k}\sum_{j\in\mathcal{C}_i} w_j (a_i + b_i z_j^*)\right] = \mathrm{E}\left[\sum_{i=0}^{k} a_i \sum_{j\in\mathcal{C}_i} w_j\right] + \mathrm{E}\left[\sum_{i=0}^{k}\sum_{j\in\mathcal{C}_i} w_j b_i z_j^*\right]$$

Comparing with Equation (2) we see that the contribution of the random assignment (the first term of the last expression) is equal:

$$\mathrm{E}\left[\sum_{i=0}^{k} a_i \sum_{j\in\mathcal{C}_i} w_j\right] = \sum_{i=0}^{k} a_i \sum_{j=1}^{m} \Pr\left[j \in \mathcal{C}_i\right] w_j = \sum_{i=0}^{k} a_i \sum_{j=1}^{m} p_i w_j = a \sum_{j=1}^{m} w_j$$

The expected value of the (almost) optimal solution of the relaxed problem is $\mathrm{E}\left[\sum_{i=0}^{k}\sum_{j\in\mathcal{C}_i} w_j b_i z_j^*\right]$. Thus it is (almost) at least the value of an optimal solution applied on the randomly restricted instance. Let the weight of an optimal solution be w_{opt} and let $\mathcal{C}_{\text{opt}}$ be the indices

Table 1: Approximation ratios α of Algorithm $A1$ (combined with a random assignment) on Max kCSP for $5 \leq k \leq 12$, with tailor-made values on s.

k	5	6	7	8	9	10	11	12
α	$2^{1.46-k}$	$2^{1.45-k}$	$2^{1.44-k}$	$2^{1.43-k}$	$2^{1.43-k}$	$2^{1.42-k}$	$2^{1.42-k}$	$2^{1.42-k}$
s	3.1	3.0	2.9	2.9	2.8	2.8	2.8	2.8

of the satisfied conjunctions. The expected value of that solution is

$$\sum_{j \in \mathcal{C}_{\mathrm{opt}}} \sum_{i=0}^{k} p_i w_j b_i = w_{\mathrm{opt}} \sum_{i=0}^{k} p_i b_i.$$

Thus, we have shown the following inequality

$$\mathrm{E}\left[\sum_{i=0}^{k} \sum_{j \in \mathcal{C}_i} w_j b_i z_j^*\right] \quad \geq \quad (1-\epsilon) \cdot w_{\mathrm{opt}} \sum_{i=0}^{k} p_i b_i,$$

where $\sum_{i=0}^{k} p_i b_i$ is the constant b appearing in Equation (2). Thus, we have obtained the same lower bound of w_{app} as we did in the previous subsection:

$$\mathrm{E}\left[w_{\mathrm{app}}\right] \geq a \cdot w_{\mathrm{tot}} + b \cdot (1-\epsilon) w_{\mathrm{opt}}.$$

5 Approximating Max kConjSAT for "small" values on k

In both Algorithm $A1$ and Algorithm $A2$ the random restriction parameter s was the same for all values of k. For specific values of k we can obtain better approximation ratios by using a larger value on s. This is the case for both algorithms. In Tables 1 and 2 the obtained approximation ratios for Max kCSP with the corresponding value on s are listed for $5 \leq k \leq 12$. For the approximation ratios in Table 2, $P = 0.72$ was used.

By comparing Table 1 and the approximation ratios mentioned in Section 3.2, which essentially is Algorithm $A1$ without random restriction, we see that even for small values of k, applying a random restriction has a positive influence for the approximation ratio.

6 Relations with probabilistical proof checking

We use the standard notation and definitions in connection with probabilistical checkable proofs, PCPs. For those unfamiliar with these, we refer to the work of Bellare et. al. [1] which also contains the history of PCPs. The complexity class $PCP_{c,s}[\log, q]$ contains

Table 2: Approximation ratios α of Algorithm $A2$ (combined with a linear relaxation with threshold rounding) on Max kCSP for $5 \le k \le 12$, with tailor-made values on s.

k	5	6	7	8	9	10	11	12
α	$2^{1.88-k}$	$2^{1.78-k}$	$2^{1.73-k}$	$2^{1.69-k}$	$2^{1.67-k}$	$2^{1.65-k}$	$2^{1.64-k}$	$2^{1.63-k}$
s	3.4	3.2	3.1	3.1	3.0	3.0	3.0	3.0

all languages that have a verifier with completeness c, soundness s, which uses only a logarithmic number of random bits and asks at most q (adaptive) questions.

The following theorem due to Trevisan shows that an approximation algorithm for Max kConjSAT implies that certain PCP classes are contained in P.

Theorem 6 (Trevisan [14]). *If* Max kConjSAT *is r-approximable for some $r \le 1$, then $PCP_{c,s}[\log, k] \subseteq P$ for any $c/s > 1/r$.*

Both algorithms presented in this work are probabilistic, thus it is not immediate that the above theorem can be applied. By inspecting the proof of Theorem 6 it is clear that only a lower bound of the value of an optimal solution is needed. For the algorithm described in Section 4 such a lower bound can be obtained deterministically from the solution of the semidefinite program in Algorithm A2 and by performing the (deterministic) linear relaxation algorithm with threshold rounding. Thus, we get the following theorem.

Theorem 7. *If Assumption 1 is true, then $PCP_{c,s}[\log, k] \subseteq P$ for any $c/s > 2^{k-1.54}$.*

We now turn to the algorithm from Section 3.2 and primarily Algorithm $A1$. As with Algorithm $A2$ we do not need to perform the rounding step (in the Feige-Goeman algorithm), because solving the semidefinite program is enough to get a lower bound of the value of an optimal solution. But the random restriction that is applied before the semidefinite relaxation has to be derandomized. Therefore, we construct a polynomial size family of k-wise independent random restrictions. As each conjunction of size k is analyzed separately we get the same approximation ratio if the random restriction is chosen from a family of k-wise independent random restrictions. By applying Algorithm $A1$ using each one of these random restrictions, it is guaranteed that at least one performs as well as the expected value over all different possible random restrictions. Thus, a lower bound on the value of the optimal solution, where the bound is at most a factor $2^{k-1.40}$ smaller than the optimal solution value is achieved.

Theorem 8. *$PCP_{c,s}[\log, k] \subseteq P$ for any $c/s > 2^{k-1.40}$.*

7 Acknowledgments

I wish to thank Johan Håstad for much appreciated help and ideas. I am also grateful to Gunnar Andersson for providing me with code that calculates the volume of a spherical tetrahedron.

References

[1] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability - towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

[2] Lars Engebretsen and Jonas Holmerin. More efficient queries in PCPs for NP and improved approximation hardness of maximum CSP. Unpublished manuscript, 2003.

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

[4] Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[5] Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM*, 42:1115–1145, 1995.

[6] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[7] Venkatesan Guruswami, Daniel Lewin, Madhu Sudan, and Luca Trevisan. A tight characterization of NP with 3 query PCPs. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 8–17, 1998.

[8] Eran Halperin and Uri Zwick. Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs. *Journal of Algorithms*, 40:185–211, 2001.

[9] Wu-Yi Hsiang. On infinitesimal symmetrization and volume formula for spherical or hyperbolic tetrahedrons. *Quarterly Journal of Mathematics (Oxford)*, 39(2):463–468, 1988.

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

[11] Sanjeev Khanna, Rajeev Motwani, Madhu Sudan, and Umesh Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1999.

[12] Sanjeev Mahajan and H. Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing*, 28(5):1641–1663, 1999.

[13] Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 191–199, 2000.

[14] Luca Trevisan. Parallel approximation algorithms by positive linear programming. *Algorithmica*, 21(1):72–88, 1998.

[15] Luca Trevisan. Approximating satisfiable satisfiability problems. *Algorithmica*, 28(1):145–172, 2000.

[16] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29:2074–2097, 2000.

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

[18] Uri Zwick. Analyzing the MAX 2-SAT and MAX DI-CUT approximation algorithms of Feige and Goemans. Manuscript, 2000.

A Approximation ratio of Algorithm $A1$

The expected approximation ratio is

$$\beta_1\alpha_1 - \rho^{-1}(1-\beta_2)\alpha_2 + \rho^{-1}p_0,$$

where $\alpha_1 = \sum_{i=1}^{5} p_i$ and $\alpha_2 = \sum_{i=2}^{5} ip_i$. Rewriting the approximation ratio as a function of p_i we get

$$\beta_1 p_1 + \sum_{i=2}^{5} p_i \left(\beta_1 - i\rho^{-1}(1-\beta_2)\right) + \rho^{-1}p_0. \tag{3}$$

We want to obtain lower bounds for $p_i(k)$. We start from the definition of $p_i(k)$:

$$\begin{aligned}
p_i(k) &= \binom{k}{i}\left(\frac{s}{k}\right)^i\left(1-\frac{s}{k}\right)^{k-i}2^{-(k-i)} \\
&= \frac{k(k-1)\dots(k-i+1)}{i!}\left(\frac{s}{k}\right)^i\left(1-\frac{s}{k}\right)^k\left(\frac{k}{k-s}\right)^i 2^i 2^{-k} \\
&= \left(1-\frac{s}{k}\right)^k\frac{(2s)^i}{i!}\frac{k(k-1)\dots(k-i+1)}{(k-s)^i}2^{-k}
\end{aligned}$$

We show that $\frac{k(k-1)\dots(k-i+1)}{(k-s)^i} > 1$ for $k \geq 6$, $1 \leq i \leq 5$ and $s = 2.56$. It is enough to show that it is true for $i = 5$.

$$\begin{aligned}
\frac{k(k-1)\dots(k-4)}{(k-s)^5} &= \frac{k(k-4)}{(k-s)^2}\cdot\frac{(k-1)(k-3)}{(k-s)^2}\cdot\frac{(k-2)}{(k-s)} \\
&= \frac{(k-2)^2-4}{(k-s)^2}\cdot\frac{(k-2)^2-1}{(k-s)^2}\cdot\frac{(k-2)}{(k-s)} \\
&> 1
\end{aligned}$$

The last inequality follows from the fact that $(k-2)^2-4 > (k-s)^2$ if $k \geq 6$ and $s = 2.56$ and all factors are thereby larger than 1.

Next we show that the factor $\left(1-\frac{s}{k}\right)^k$ increases with larger k.

Claim 9. *$\left(1-\frac{s}{k}\right)^k$ is a monotone increasing function of k for $k > s$.*

Proof. We prove the claim by showing that the derivative is positive for all $k > s$.

$$\frac{\partial}{\partial k}\left(1-\frac{s}{k}\right)^k = \left(1-\frac{s}{k}\right)^k\left(\ln(1-\frac{s}{k}) + \frac{s}{k-s}\right)$$

Table 3: Values of $p_i(k)$ when $k \to \infty$ and for the lower bound $\hat{p}_i$ of $p_i(k)$, for all $k \geq k_0 = 1000$.

i	0	1	2	3	4	5
$p_i \cdot 2^k$	0.0773	0.3958	1.0132	1.7293	2.2135	2.2666
$\hat{p}_i \cdot 2^k$	0.0771	0.3945	1.0099	1.7236	2.2062	2.2592

As $\left(1 - \frac{s}{k}\right)^k$ is positive for all $k > s$ we need only to show that $\ln(1 - \frac{s}{k}) + \frac{s}{k-s}$ is positive.

$$\begin{aligned}
\ln(1 - \frac{s}{k}) + \frac{s}{k-s} &= \ln \frac{k-s}{k} + \ln e^{s/(k-s)} \\
&> \ln \frac{k-s}{k} + \ln(1 + \frac{s}{k-s}) \\
&= \ln \frac{k-s}{k} + \ln \frac{k}{k-s} \\
&= 0,
\end{aligned}$$

where we used the fact that $e^t > 1 + t$. $\square$

Thus a lower bound $\hat{p}_i$ of $p_i(k)$ for all $k \geq k_0 \geq 6$ is

$$\hat{p}_i = \left(1 - \frac{s}{k_0}\right)^{k_0} \frac{(2s)^i}{i!} 2^{-k}.$$

In the expression for the approximation ratio, Equation (3), all the coefficients of p_i, for $0 \leq i \leq 5$, are positive, otherwise conjunctions of length i would not have been included. Setting $k_0 = 1000$ and substituting the p_i values for the lower bounds $\hat{p}_i$ in Equation (3) we obtain a lower bound of $2^{1.40-k}$ for the approximation ratio of Max kConjSAT for all $k \geq k_0$. The lower bounds, $\hat{p}_i$, are shown in Table 3 together with $p_i(k)$ when $k \to \infty$. By use of computer we have shown that for smaller values, $5 \leq k < 1000$, our algorithm approximates Max kConjSAT within $2^{1.40-k}$ of the optimum.[2]

- *Author's comment: Put computer program on the web. Johan: you can find it at* `~ghast/Forskning/Matlab/APPpapper/k5to1000.m`. -

[2]The matlab program is available at `http://www.nada.kth.se/~ghast/research/k5to1000.m`.