The Hardness of Approximation:
Gap Location
(Preliminary Version)

by
E. Petrank

Technical Report #754
December, 1992
The Hardness of Approximation: Gap Location

(Preliminary Version)

Erez Petrank *
Computer Science Department
Technion - Israel Institute of Technology
Haifa, Israel

Abstract

We refine the complexity analysis of approximation problems, by relating it to a new parameter called gap location. Many of the results obtained so far for approximations yield satisfactory analysis also with respect to this refined parameter, but some known results (e.g. MAX-k-COLORABILITY, MAX 3-DIMENSIONAL MATCHING and MAX NOT-ALL-EQUAL 3SAT) fall short of doing so. A second contribution of our work is in filling the gap in these cases by presenting new reductions.

Next, we present definitions and hardness results of new approximation versions of some NP-Complete optimization problems. The problems we treat are VERTEX COVER (for which we define a different optimization problem from the one treated in [PY91]), k-EDGE COLORING, SET SPLITTING and a restricted version of FEEDBACK VERTEX SET and FEEDBACK ARC SET.

Our last contribution regards two natural problems in coding theory. We show that these two problems are NP-Hard to approximate.

* email: erez@cs.technion.ac.il. Supported by grant no. 89-00312 from the United States — Israel Bina
tional Science Foundation, Jerusalem, Israel
1 Introduction

We begin by presenting a partition of the class of optimization problems into two categories. We later introduce the notion of gap location which refers to the second category.

1.1 Two Categories of Optimization Problems

Generally, an optimization problem consists of a set of instances and a function \( f : \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{R} \) that assigns, for each instance \( I \) and each candidate solution \( \sigma \), a real number \( f(I, \sigma) \) called the value of the solution \( \sigma \). The optimization task is to find a solution \( \sigma \) to a problem instance \( I \), such that \( f(I, \sigma) \) is the largest possible over all \( \sigma \in \{0,1\}^* \). We say that an algorithm \( A \) approximates a maximization (resp. minimization) problem \( \Pi \) to within \( 1 - \epsilon \) (resp. \( 1 + \epsilon \)) if for every instance \( I \) of \( \Pi \), whose optimal solution has value \( OPT(I) \), the output of \( A \) on \( I \) satisfies \( (1 - \epsilon)OPT(I) \leq A(I) \leq OPT(I) \) (resp. \( OPT(I) \leq A(I) \leq (1 + \epsilon)OPT(I) \)). Most natural optimization problems are associated with a decision problem in NP. We have a relation \( R \subseteq \{0,1\}^* \times \{0,1\}^* \) which is checkable in polynomial time (i.e., given \( I, \sigma \) it is possible to check in time polynomial in \( |I| \) whether \( (I, \sigma) \in R \), and we call \( \sigma \) a valid solution to an input \( I \) if \( (I, \sigma) \in R \). The decision problem is whether there exists a valid solution to the input \( I \). Two natural categories of corresponding optimization problems follow:

1. "The Largest Solution"

Here, we associate valid solutions with some natural “size” which we would like to maximize/minimize. More formally, we are trying to maximize the function

\[
f(I, \sigma) = \begin{cases} 
\text{size}(\sigma) & \text{if } (I, \sigma) \in R \\
-\infty & \text{otherwise}
\end{cases}
\]

where \( \text{size}(\cdot) \) is a function which depends on the problem and can be efficiently extracted from \( \sigma \) (usually \( \text{size}(\sigma) \) is the number of elements encoded in the solution \( \sigma \)). Two examples in this category are MAX CLIQUE, in which we are looking for the size of the largest clique in the input graph, and MIN COLORING, in which we would like to find the minimum number of colors required to color the input graph such that no two adjacent vertices have the same color.

2. "The Quality of the Solution"

Here, we assume that the condition \( (I, \sigma) \in R \) contains a large (yet polynomially bounded) number of natural “sub-conditions” and our task is to find the maximum number of sub-conditions that can be satisfied by a single (i.e., the best) solution. In this category, we have MAX-SAT, in which we are trying to find the maximum number of clauses that can be satisfied by an assignment to the input formula, and MAX 3-COLORABILITY, in which we are trying to find the maximum number of consistent edges in a 3-coloring of the input graph. Note that, in this setting, the best solution of an instance \( I \) is not necessarily a valid solution, since there are instances that do not

---

1 We replace \(-\infty\) by \(+\infty\) when a minimization problem is involved.

2 We call an edge in a graph \( G \) consistent regarding a 3-coloring of \( G \), if its two adjacent vertices are assigned different colors.
have a valid solution. Also, A solution \( \sigma \) is a valid solution of an instance \( I ((I, \sigma) \in R) \) iff \( \sigma \) satisfies all sub-conditions implied by \( I \).

Approximation problems can be partitioned in the same manner and so can results concerning the difficulty of approximations. The first category contains problems such as approximating the size of the biggest clique in the input graph (the hardness of this problem was shown in [FGLSS91, AS92, ALMSS92]), or the minimum number of colors needed to properly color the input graph (see [LY92]). The second category includes problems such as approximating the maximum number of clauses that can be simultaneously satisfied in an input formula (see [ALMSS92]), or approximating the maximum number of consistent edges in a (best) 3-coloring of the input graph (see [PY91]).

The gap location parameter is a natural measure which arises when analyzing the hardness of approximation problems in the second category.

### 1.2 The Gap-Location parameter

Practically all researchers in the area noticed the connection between the hardness of approximating a problem and the existence of a "gap" that is hard to differentiate. For example, the hardness of approximating MAX-SAT was shown by proving that there exists a constant \( \varepsilon_0 > 0 \) such that, unless \( P=NP \), one can not distinguish in polynomial time between formulae that all their clauses can be satisfied and formulae that only a fraction of \( 1 - \varepsilon_0 \) of their clauses can be satisfied [ALMSS92]. The \( L \)-reductions, that were used in [PY91], preserve the existence of such gaps and, thus, a polynomial time inseparable gaps appear in all MAX-SNP-Hard problems. This implies that these problems have no polynomial time approximation schemes. Loosely speaking, a hard gap for an optimization problem \( \Pi \) consists of two reals \( 0 \leq \alpha_0, \varepsilon_0 \leq 1 \) such that, given an instance \( I \) of \( \Pi \), it is \( NP \)-Hard to tell whether \( I \) has a solution that satisfies at least a fraction of \( \alpha_0 \) of the sub-conditions posed by \( I \) or whether any solution of \( I \) satisfies at most a fraction of \( \alpha_0 - \varepsilon_0 \) of the sub-conditions posed by \( I \). Such a hard gap is said to have location \( \alpha_0 \). For example, the hard gap proven for MAX-SAT has location 1. Let us first argue that the "location" of the hard gap shown for MAX-SAT is the "right" one and then explain why there are some problems whose proofs of hardness are weaker in this respect.

Our main interest lies in the original question of satisfiability, i.e., in telling whether a formula \( \varphi \) has an assignment that satisfies all its clauses or not. It is therefore interesting to see that we cannot solve even the easier question of whether all the clauses of \( \varphi \) can be satisfied (simultaneously) or whether any assignment to \( \varphi \) satisfies at most a fraction of \( 1 - \varepsilon_0 \) of its clauses. It would be somewhat artificial (and clearly of lesser interest) to show that it is impossible to tell whether a formula \( \varphi \) has an assignment that satisfies more than \( \frac{2}{3} \) of its clauses or whether no assignment to \( \varphi \) satisfies more than a fraction of \( \frac{2}{3} - \varepsilon_0 \) of its clauses, although showing this would still imply the hardness of approximating MAX-SAT (i.e., approximating the maximum number of simultaneously satisfied clauses). Furthermore, the intuition about the "right" location of the hard gap coincides with the power of such a result. Namely, for MAX-SAT, the existence of a hard gap at location 1 implies the existence of a hard gap at any location \( \frac{1}{2} < \alpha \leq 1 \). That is, the fact that one cannot tell in polynomial time whether a formula has an assignment that satisfies all its clauses or whether all assignments to \( \varphi \) satisfy at most a fraction of \( 1 - \varepsilon_0 \) of its clauses, implies the fact that one
cannot tell in polynomial time whether a formula has a satisfying assignment that satisfies more than a fraction of $\alpha$ of its clauses or whether any assignment to $\varphi$ satisfies at most a fraction of $\alpha - \epsilon_0$ of its clauses, where $\epsilon_0$ is a constant that depends only on $\epsilon_0$ and $\alpha$. There are no hard gaps at locations $0 \leq \alpha \leq \frac{1}{2}$, since any formula has an assignment that satisfies at least half of its clauses. We conclude that, in this respect, the proof of the hardness of approximating MAX-SAT is the strongest possible.

We conjecture a generalization of this example. Suppose we have a *natural* optimization problem that seeks the quality of the best solution. We conjecture that showing a hard gap at location $\alpha_0 = 1$ can be used to prove the existence of hard gaps in all other locations (in which they exist).

In previous works, hard gaps are not always shown at location $\alpha_0 = 1$. For example, let us return to MAX $k$-COLORABILITY. Recall that, in MAX $k$-COLORABILITY, we look for the maximum number of consistent edges in a $k$-coloring of the input graph. The interesting original problem (for $k \geq 3$) is to tell whether a graph $G(V, E)$ is $k$-colorable or not. Relaxing the precision requirement, we would like to know if it is easier to tell whether $G$ has a $k$-coloring, for which all $|E|$ edges in $G$ are consistent, or whether for all $k$-colorings of $G$ at most $(1 - \epsilon)|E|$ edges are consistent. Can this relaxation be determined in polynomial time for any constant $\epsilon$? This question was not considered before. Instead, following the gap-preserving L-reductions in [PY91], we get that there are constants $0 < \epsilon_0, \alpha_0 < 1$ such that, unless $P=NP$, it is not possible to tell in polynomial time whether a given graph $G(V, E)$ has a $k$-coloring with $\geq \alpha_0|E|$ consistent edges, or whether for any $k$-coloring of $G$ at most $(\alpha_0 - \epsilon_0)|E|$ edges are consistent. Using this hard gap we can indeed say that there is no polynomial time approximation scheme for MAX $k$-COLORABILITY unless $P=NP$, but the hardness of the interesting gap (i.e., $\alpha_0 = 1$) remains open.

The importance of the gap at location $\alpha_0 = 1$ can be also expressed in terms of the analogue search problem. The implication of [PY91] is that given a graph $G$, we cannot find a $k$-coloring of $G$ that is as close as desired to the optimal solution (unless $P=NP$). But, suppose we are given a $k$-colorable graph and we would like to color it such that as many edges as possible are consistent. [PY91] gives no evidence that we cannot achieve this task in polynomial time, such that the number of consistent edges is greater than $(1 - \epsilon)|E|$ for any constant $\epsilon$.

### 1.3 Summary of Results

The hardness of finding a $k$-coloring that has "almost" as many consistent edges as possible, for $k$-colorable graphs, is implied by our showing a hard gap at location 1 for MAX $k$-COLORABILITY (for all $k \geq 3$). We thus settle the problem raised in the previous subsection. We use a different reduction (than the one in [PY91]). Two other problems that were previously shown hard to approximate using a gap location different than 1 are 3-DIMENSIONAL MATCHING [Kan91] and NOT-ALL-EQUAL-3SAT [PY91]. We strengthen these results by showing that these problems indeed possess a hard gap at location 1.

Next, we define new approximation versions of some known NP-Complete problems,

---

3This can be shown by a padding argument. Namely, use enough new variables $y_1, \ldots, y_l$ and add the clauses $(y_i)$ and $\overline{(y_i)}$, $1 \leq i \leq l$, to the original formula.
in the spirit of approximating the quality of the best solution. In particular, we define approximation versions of:

- **VERTEX COVER [Kar72]**. Note that the version that was treated in [PY91] is in the spirit of the largest solution. We treat the version that seeks the quality of the best solution.

- **k-EDGE COLORING (CHROMATIC INDEX) [Hoy81, LG83]**. In this case, it is not hard to approximate the size of the smallest solution, since there is a polynomial time algorithm that colors the edges of a graph of degree $k$, with $k + 1$ colors (Vizing’s Theorem, see [Ber73]). Again, we treat the version that seeks the quality of the best solution (i.e., a best $k$ edge-coloring of a $k$-degree graph).

- **SET SPLITTING [Lov73]** and

- restricted versions of FEEDBACK VERTEX SET and FEEDBACK ARC SET [Kar72].

We show that all these problems possess a hard gap at gap-location 1.

### Results in Coding Theory

Our last contribution concerns an important (NP-Complete) problem in coding theory - the decoding of linear codes. Suppose we have a (linear) code $C$, and a word $w$ which is a code-word that was distorted by some erroneous channel. By the principle of maximum likelihood decoding, we would like to find a code-word $c_w$ which is “closest” to $w$ in the code. This is a code-word that agrees with $w$ in as many bits as possible. The problem of finding the exact number of distorted bits in $w$ was shown NP-Complete by Berlekamp, McEliece and van Tilborg [BMT78]. We use a different reduction to show that unless P=NP, there is no polynomial time algorithm that approximates the number of undistored bits in $w$ (i.e., the number of entries in which $w$ and $c_w$ agree) to within a ratio of $1 - \epsilon$ for some $\epsilon > 0$. This problem is a natural member in the category of approximating the quality of the best solution. Here, a solution is a code-word and its quality is its closeness to $w$.

Using the same techniques, we are also able to treat another important problem in coding theory (also shown NP-Complete in [BMT78]). Here, we are given a code $C$ and an integer $h$, and we want to tell whether there exists a code-word of weight $h$. The approximation problem that naturally arises is determining the existence of a code-word that is “close” to the given weight $h$. The hardness of this problem is implied by our showing that, given a code, it is NP-Hard to approximate the Hamming weight of the “heaviest” code-word within a ratio of $1 - \epsilon$ for some $\epsilon > 0$.

### 2 The Gap Location Parameter

In this section, we would like to introduce the definitions concerning hard gaps. We follow the definitions with an informal discussion and two conjectures. We consider only optimization problems that seek the quality of the best solution. Denote by $N(I)$ the number of sub-conditions posed by the instance $I$. $OPT(I)$ denotes the number of sub-conditions which are satisfied by the best solution of $I$ (i.e., the solution that maximizes the number of satisfied
sub-conditions). Note that the main concern of this paper is the hardness of approximating \( \text{OPT}(I) \) whereas \( N(I) \) can be efficiently extracted from \( I \).

**Definition 2.1 (The parameters of a hard gap):** Consider an optimization problem \( \Pi \) in which we are looking for the quality of the best solution. Suppose that it is NP-Hard to tell whether \( \text{OPT}(I) \leq \alpha_1 \cdot N(I) \) or whether \( \text{OPT}(I) \geq \alpha_0 \cdot N(I) \), given that \( I \) satisfies one of these conditions. Then we say that there exists a hard gap for the problem \( \Pi \), at location \( \alpha_0 \), with width \( \alpha_0 - \alpha_1 \).

In this work we are only concerned with gaps of constant width. In the sequel, we omit the width parameter. Throughout the paper, we show that various problems have hard gaps. Let us state (the obvious) implication of hard gaps on the hardness of approximations.

**A General Remark:** If a maximization (resp. minimization) problem \( \Pi \) has a hard gap at any location \( 0 < \alpha \leq 1 \) then there exists a constant \( \epsilon > 0 \) such that, unless \( P=NP \), there is no polynomial time algorithm that approximates \( \Pi \) to within \( 1 - \epsilon \) (resp. \( 1 + \epsilon \)).

The instances of an optimization problem always satisfy \( 0 \leq \text{OPT}(I) \leq N(I) \), but sometimes the solution interval is more restricted:

**Definition 2.2 (The solution interval):** We say that an optimization problem, in which we seek the quality of the best solution, has solution interval \( [\beta_1, \beta_2] \) if \( \liminf_{|I| \to \infty} \frac{\text{OPT}(I)}{N(I)} = \beta_1 \) and \( \limsup_{|I| \to \infty} \frac{\text{OPT}(I)}{N(I)} = \beta_2 \).

Namely, neglecting a finite number of instances, \( \text{OPT}(I) \) satisfies \( \beta_1 \cdot N(I) \leq \text{OPT}(I) \leq \beta_2 \cdot N(I) \). Usually, \( \beta_2 = 1 \) in natural problems.

**Informal discussion**

Let us restrict the discussion to (natural) problems that seek the quality of the best solution and that are hard to approximate (i.e., have a hard gap somewhere in their solution interval). Also, let the solution interval of \( \Pi \) be \( [\beta_\Pi, 1] \). We would like to state two informal conjectures concerning the issue of gap-location. The first conjecture concerns the power of a hard gap at location 1. We conjecture that the existence of a hard gap at location 1 can be used to prove the existence of hard gaps at all locations in \( (\beta_\Pi, 1] \). A second conjecture is that a problem that is hard to approximate has hard gaps almost at all locations of its solution interval. Namely, if a problem has a hard gap then it has hard gaps either at all locations in \( (\beta_\Pi, 1) \) or at all locations in \( (\beta_\Pi, 1] \). We proceed by making a clear distinction between these two possibilities.

Consider the following decision problem related to an approximation problem \( \Pi \). Given an instance \( I \), determine whether \( \text{OPT}(I) = N(I) \). For example, the decision problem related to MAX-SAT is SAT (i.e., given a formula \( \varphi \), determine whether all its clauses can be satisfied). We partition the optimization problems, which are hard to approximate, into two classes according to the difficulty of their related decision problems.\(^4\) The first class contains optimization problems for which it is NP-Hard to determine whether an input \( I \) has \( \text{OPT}(I) = N(I) \). This class contains problems as MAX-SAT (proven hard to approximate in

\(^4\)We do not claim that all decision problems are either NP-Hard or easy, but practically all known interesting problems, of this form, do fall into one of these two categories.
contains problems for which it is easy to decide whether an input $I$ has $OPT(I) = N(I)$. This class contains problems such as MAX-CUT (which can be also viewed as MAX 2-COLORABILITY), MAX-2SAT (both shown hard to approximate in [PY91]), and MAX-DECODE (see §3.1). We believe that any natural problem of the first class has hard gaps at all locations $\alpha \in (\beta_1, 1]$ and that any natural problem of the second class has hard gaps at all locations $\alpha \in (\beta_1, 1)$. Note that we can never have a hard gap at location $\beta_1$, since we know that all instances have $OPT(I) \geq \beta_1 \cdot N(I)$ and none has $OPT(I) \leq (1 - \epsilon)\beta_1 \cdot N(I)$, for any $\epsilon > 0$.

Let us also explain our first conjecture. We believe that, given a hard gap at location 1, one can use padding argument to "transfer" the hard gap to any other location in the solution interval $(\beta, 1]$. This can be demonstrated on the problems MAX-SAT (see a footnote in the introduction) and MAX $k$-COLORABILITY. However, this method does not enable us to prove a hard gap at location 1 using a hard gap at location $\alpha_0 \in (\beta_1, 1)$. In fact, a general claim that a hard gap at location $\alpha_0 \in (\beta_1, 1)$ implies a hard gap at location 1 must be false, since there are problems (all problems in the second class) for which there exist a hard gap at some location $\alpha_0 \in (\beta_1, 1)$ but there does not exist a hard gap at location 1.

3 Results concerning Coding Theory

In this section we consider two problems in coding theory: The decoding of linear codes and the general problem of finding the weights of a code. We show that these problems are hard to approximate. For details and motivation see [MS81].

3.1 MAX-DECODE is hard to approximate

Consider the following problem. We are given a word $v$ and a generating matrix $M$ of a code $C$. Let $c_v$ be a nearest code-word to $v$. Find the number of entries (bits) in which $v$ and $c_v$ agree. We prove that it is NP-Hard to approximate this number to within $1 - \delta$, for some constant $\delta > 0$. Note that in this problem we are trying to approximate the quality ("closeness") of the best ("closest") solution (code-word). Also note that there is no hard gap at location $\alpha = 1$ for this problem, since it is easy to tell whether a given word $v$ is a code-word or not. We show a hard gap at location 1/2 and one may use padding arguments to "move" the hard gap to any location in (0, 1). Let us begin with the definition.

Definition 3.1 The problem MAX-DECODE:
Input: A word $v \in \{0, 1\}^l$ and an $n \times 1$ matrix $M$, that generates a linear code denoted $C$.
Problem: Find the number of entries in which $v$ agrees with its nearest code-word.

Theorem 3.2 The problem MAX-DECODE possesses a hard gap at location 1/2.

Proof: We shall use a reduction from MAX-3SAT (for definition see Appendix A). Given a 3-CNF formula $\varphi$, with $n$ variables and $m$ clauses, we build a binary matrix $M^\varphi$ of dimension $n \times (8m)$ and a vector $v^\varphi \in \{0, 1\}^{8m}$. Regard $M^\varphi$ as a generating matrix of a code $C^\varphi$. We

---

[5] Add a large enough clique to the graph, and do not connect it to any original vertex. Note that the solution interval here is $[\frac{1}{2}, 1]$ since any graph has a $k$-coloring for which $\frac{1}{k} \cdot |E|$ edges are consistent.
prove that the "best" truth assignment to \( \varphi \) satisfies \( \delta m \) clauses in \( \varphi \) iff the nearest code-word in \( C^\varphi \) to the word \( v^\varphi \) agrees with \( v^\varphi \) on \( 4\delta m \) bits. Since there exists an \( \epsilon \) such that it is NP-hard to determine whether an input formula \( \varphi \) is satisfiable or whether all truth assignments to \( \varphi \) satisfy at most a fraction of \( 1 - \epsilon \) of its clauses, it follows that it is NP-Hard to tell, given a code \( C \) and a word \( v \), whether there exists a code-word that agrees with \( v \) on half of its entries or whether all code-words agree with \( v \) on at most a fraction of \( \frac{1}{2} - \frac{\epsilon}{2} \) of its entries. Thus, we have a hard gap for MAX-DECODE at location 1/2.

Intuitively, we associate possible code-words with possible truth assignments by relating them both to vectors in \( \{0,1\}^n \). We relate a truth assignment (in \( \{FALSE, TRUE\}^n \)) with a vector in \( \{0,1\}^n \) by identifying TRUE with 1 and FALSE with 0. Recall that a code-word is a linear combination of the \( n \) rows in \( M^\varphi \). We relate a code-word \( v \) with the vector \( \alpha \in \{0,1\}^n \) that satisfies \( v = \alpha \cdot M^\varphi \). In other words, we associate the \( i \)-th row in \( M^\varphi \) with the variable \( x_i \) in \( \varphi \), and we get the code-word \( w_i \) (associated with the truth assignment \( \tau \)) by adding (in \( GF[2] \)) the rows of the variables that are assigned TRUE by \( \tau \).

The construction of \( M^\varphi \) is done clause by clause. For each clause we allocate eight columns in the matrix \( M^\varphi \) and eight corresponding entries in the vector \( v^\varphi \). In this way, we construct a "narrow" matrix of dimension \( n \times 8 \) for each clause and a "short" vector of length eight that corresponds to it. We construct this sub-matrix and sub-vector, such that if a truth assignment \( \tau \) satisfies the clause \( C_j \) then the linear combination, implied by \( \tau \) on this sub-matrix, agrees with the sub-vector on exactly four bits, while if \( \tau \) does not satisfy \( C_j \) then the linear combination, implied by \( \tau \), disagrees with the sub-vector on all eight bits. In this sub-matrix we fill only the three rows that correspond to the variables of the clause. The other rows are filled with zeros. We need to build three rows (of length 8) such that any linear combination of these three rows, except one specified combination, has Hamming distance 4 from a fourth row (which will be assigned to the vector). The one exceptional combination, which corresponds to an assignment that doesn't satisfy the clause, has distance 8 from the fourth row. These rows have to be constructed according to the signs (negations) of the three variables in \( C_i \), since these negations determine which truth assignment does not satisfy the clause. We use one of eight basic blocks to fill these rows, according to the possible signs of the variables in the clause. A possible construction of such basic blocks appears in §3.3.

After constructing a sub-matrix (of dimension \( n \times 8 \)) for each clause, we concatenate all the sub-matrices to get the matrix \( M^\varphi \) of dimension \( n \times (8m) \). We also concatenate all the sub-vectors correspondingly, to get the vector \( v^\varphi \) of length \( 8m \). Now, summing over all clauses, we get that each truth assignment \( \tau \) of \( \varphi \), that satisfies \( \delta m \) of the clauses, induces a linear combination of \( M^\varphi \) (i.e., a code-word \( w_\tau \)) which has a Hamming distance of exactly \( 4 \times \delta m + 8 \times (1 - \delta)m \) from \( v^\varphi \).

The Reduction - Formal description

Formally, we denote the eight basic blocks by \( G[s_1,s_2,s_3] \), for \( s_1, s_2, s_3 \in \{0,1\} \). Each such basic block is a matrix of dimension \( 4 \times 8 \) with the following property. Let \( \beta \in \{0,1\}^4 \), \( \beta \neq (0,0,0,0) \), then:

\[
H(\beta \cdot G[s_1,s_2,s_3]) = \begin{cases} 
8 & \text{if } \beta = (s_1,s_2,s_3,1) \\
4 & \text{otherwise}
\end{cases}
\]
A possible construction of such basic blocks appears in Subsection 3.3.

Suppose the input to the reduction is a formula $\varphi = \bigwedge_{i=1}^{m} C_i$. In the sequel, we identify TRUE with 1 and FALSE with 0. Denote by $(s_{i_1}, s_{i_2}, s_{i_3}) \in \{0, 1\}^3$ the assignment to the variables of $C_i$ that does not satisfy the clause $C_i$. For each clause $C_i$, $1 \leq i \leq m$, we consider the columns between $8i - 7$ and $8i$ in the matrix $M^\varphi$. We assign zeros to all the rows in these columns except the rows $i_1, i_2$ and $i_3$. We assign to these three rows the top three rows of the basic block $G[s_{i_1}, s_{i_2}, s_{i_3}]$. The last row of the basic block is assigned to the places between $8i - 7$ and $8i$ in the vector $v^\varphi$.

Validity of the Reduction

Let us state and prove the main lemma.

Lemma 3.3 Let $(M^\varphi, v^\varphi)$ be the output of our reduction on a 3-CNF formula $\varphi$, and let $\alpha \in \{0, 1\}^n$ be an assignment to $(x_1, x_2, ..., x_n)$. If the number of clauses in $\varphi$ that are satisfied by the assignment $\alpha$ is $\delta m$, $0 \leq \delta \leq 1$, then $\alpha M^\varphi$ agrees with $v^\varphi$ on $4m\delta$ entries.

Proof: Let $i$ be an integer $1 \leq i \leq m$. Consider the sub-matrix $M_i^\varphi$ of $M^\varphi$ that consists of the columns of $M^\varphi$ in the range between $8i - 7$ to $8i$. Also, consider the sub-vector $v_i$ that consists of the elements of $v$ in the places between $8i - 7$ to $8i$. We claim that

$$H(\alpha M_i^\varphi + v_i^\varphi) = \begin{cases} 4 & \text{if } \alpha \text{ satisfies } C_i \\ 8 & \text{otherwise} \end{cases} \quad (1)$$

To see that Equation (1) is true, recall that all the rows in $M_i^\varphi$ are full of zeros except for $i_1, i_2$ and $i_3$. Also, recall that the assignment $\alpha$ does not satisfy clause $C_i$ iff $(\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}) = (s_{i_1}, s_{i_2}, s_{i_3})$. By the construction of $M^\varphi$ and $v^\varphi$ in the columns between $8i - 7$ and $8i$, we are left exactly with the basic block $G[s_{i_1}, s_{i_2}, s_{i_3}]$. That is,

$$\alpha M_i^\varphi + v_i^\varphi = (\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}, 1) \cdot G[s_{i_1}, s_{i_2}, s_{i_3}]$$

By the properties of our basic block, we get Equation (1) as needed.

Now, summing over all $1 \leq i \leq m$, we get

$$H(\alpha M^\varphi + v^\varphi) = \sum_{i=1}^{m} H(\alpha M_i^\varphi + v_i^\varphi)$$

Since the number of clauses in $\varphi$ that are satisfied by the assignment $\alpha$ is $\delta m$, we can use Equation (1) to get:

$$\delta m \cdot 4 + (1 - \delta)m \cdot 8 = 4m(2 - \delta)$$

as we need. ■

3.2 The maximum weight of a code is hard to approximate

Consider the problem of finding the maximum Hamming weight over all code-words. Namely, we are given a matrix $M$ of dimension $l \times k$ and we have to determine the maximum of $H(\alpha M)$ (the Hamming weight of $\alpha M$) over all $\alpha \in \{0, 1\}^l$. We show that this problem has a hard gap. Note that the hardness of this problem implies the hardness of the approximation
version of an important problem in coding theory. Namely, given a code $C$ and an integer $h$, determine whether there exists a word, the weight of which, is “close” to $h$. This problem can be regarded as seeking the quality of the best solution by noting that we are looking for the quality (i.e. the number of entries that are 1) of the best solution (code-word). Note that this problem has no hard gap at location 1, since it is easy to tell whether $(1,1,...,1)$ is a code-word or not. The location of the gap we show is $4/7$. Again, one can use a padding argument to translate this hard gap to hard gaps at all locations in $(0,1)$.

Definition 3.4 The problem MAX-WEIGHT:
Input: a generating matrix $M$ of a code $C$.
Problem: Find the maximum, over all code-words $v \in C$, of the Hamming weight of $v$.

Theorem 3.5 The problem MAX-WEIGHT possesses a hard gap at location $4/7$.

Proof: We reduce MAX-3SAT to MAX-WEIGHT in a similar reduction to the reduction of §3.1. Given a 3-CNF formula $\varphi$, with $n$ variables and $m$ clauses, we build a matrix $M_\varphi$ of dimension $(n+1) \times 56m$. We show that the best truth assignment to $\varphi$ satisfies $(1-\delta)m$ clauses in $\varphi$ iff there exists a code-word $w_\varphi$ of Hamming weight $(8-\delta)4m$. Thus, we get a hard gap for MAX-WEIGHT at location $4/7$.

Here, we associate a code-word with a truth assignment in the following way. Let $\tau$ be a truth assignment to $\varphi$ in $\{0,1\}^n$. The code-word $v_\tau$ is defined as $(\tau,1) \cdot M_\varphi$. In other words, we associate the $i$-th row, $1 \leq i \leq n$, in the generating matrix $M_\varphi$ with the variable $x_i$ and we get $v_\tau$ by adding all the rows in $M_\varphi$ whose variables are assigned TRUE by $\tau$ and the last row (the $(n+1)$-th row). The justification to always adding the last row (in the analysis of this reduction) is that adding the last row cannot decrease the weight of the sum, and that we are interested in the heaviest code-word.

For each clause $C_i$, we build a sub-matrix of dimension $(n+1) \times 56$. For this sub-matrix, it holds that for each truth-assignment $\tau$, if $\tau$ satisfies $C_i$ then the linear combination implied by $\tau$, on this sub-matrix, has Hamming weight $28+4$ and if $\tau$ does not satisfy $C_i$ then the linear combination implied by $\tau$ has Hamming weight 28. In this sub-matrix, we fill the rows that correspond to the three variables of $C_i$ and the last row. The other rows are filled with zeros. We use seven out of the eight basic basic blocks (of §3.3) to fill these $7 \cdot 8$ columns and four rows.

Formally, we deal with each clause separately. For the clause $C_i$, $1 \leq i \leq m$, we consider only the columns between $56i-55$ and $56i$ in the matrix $M_\varphi$, and only the rows $i_1,i_2,i_3$ and $n+1$. The rest of the matrix is filled with zeros. Let $(s_{i_1},s_{i_2},s_{i_3}) \in \{0,1\}^3$ be the assignment to the three variables of $C_i$ that does not satisfy $C_i$. We assign to the four rows and 56 columns, which are associated with clause $C_i$, the seven basic blocks - $G[\alpha_1,\alpha_2,\alpha_3]$ for $\alpha_1,\alpha_2,\alpha_3 \in \{0,1\}$ and $(\alpha_1,\alpha_2,\alpha_3) \neq (s_{i_1},s_{i_2},s_{i_3})$. Recall that $G[\alpha_1,\alpha_2,\alpha_3]$ satisfies for any for any $\beta \in \{0,1\}^4$, $\beta \neq (0,0,0,0)$:

$$H(\beta \cdot G[s_1,s_2,s_3]) = \begin{cases} 8 & \text{if } \beta = (s_1,s_2,s_3,1) \\ 4 & \text{otherwise} \end{cases}$$

In order to prove that the construction has the needed properties, let us (again) view the assignment as a vector in $\{0,1\}^n$ and show two things:
1. For all \( \alpha \in \{0,1\}^n \), \( \alpha \) satisfies \( 6m \) clauses in \( \varphi \) iff the code-word \((\alpha,1) \cdot M_\varphi\) is of Hamming weight \((7 + \delta) \cdot 4m\).

2. For all \( \alpha \in \{0,1\}^n \), the code-word \((\alpha,0) \cdot M_\varphi\) is of Hamming weight at most \(7 \cdot 4m\).

To see that the first condition is true, we consider an assignment \( \alpha \) and a clause \( C_i, 1 \leq i \leq m \). If the clause is not satisfied by \( \alpha \), then, by the property of the basic blocks, all its seven basic blocks, contribute exactly Hamming weight 4 to the code-word \((\alpha,1) \cdot M_\varphi\), while if the clause is satisfied, then six of its basic blocks contribute Hamming weight 4 and one basic block contributes Hamming weight 8 to the code-word \((\alpha,1) \cdot M_\varphi\). Summing over all clauses, we get that the Hamming weight of the code-word \((\alpha,1) \cdot M_\varphi\) is exactly \((7 + \delta) \cdot 4m\).

The second condition is true, since for each of the \(7m\) basic blocks in the matrix, we have a linear combination that does not include the last row. By the property of our basic block, any linear combination of the three upper rows has either Hamming weight 4 (any non-trivial combination) or 4 (any non-trivial combination). Summing over all \(7m\) basic blocks in \(M_\varphi\), we get that the maximum Hamming weight possible is \(4 \cdot 7m\) as required.

### 3.3 The Basic Blocks

Let us show a possible construction of the basic blocks that are needed for the reduction from MAX-3SAT to MAX-DECODE and to MAX-WEIGHT. We need four binary vectors of length 8, such that all linear combinations of the first three rows, except one, have Hamming distance 4 from the fourth row. The one exceptional combination has distance 8 from the fourth row. In addition, all linear combinations of the first three rows have Hamming weight at most 4.

Formally, for any \( s_1, s_2, s_3 \in \{0,1\} \), we need to construct a binary matrix of dimension \(4 \times 8\) which satisfies for any \( \beta \in \{0,1\}^4 \), \( \beta \neq (0,0,0,0)\):

\[
H(\beta \cdot G[s_1, s_2, s_3]) = \begin{cases} 
8 & \text{if } \beta = (s_1, s_2, s_3, 1) \\
4 & \text{otherwise}
\end{cases}
\]

**Construction:**

The three upper 3 rows in \(G[s_1, s_2, s_3]\), for any \( s_1, s_2, s_3 \in \{0,1\} \), are:

\[
B = \begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

The last row of \(G[s_1, s_2, s_3]\) is set to

\[(s_1, s_2, s_3) \cdot B + (1,1,1,1,1,1,1,1)\]

For example:

\[
G[1,1,0] = \begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]
Validity of construction:

An important property of matrix $B$ is that any linear combination of its rows (except the trivial one) yields a vector with Hamming weight exactly 4. Namely, for all $\alpha \in \{0,1\}^3$ $\alpha \neq (0,0,0)$ we have

$$H(\alpha \cdot B) = 4$$

(2)

In other words, for any $s_1, s_2, s_3 \in \{0,1\}$ and any $\alpha \in \{0,1\}^3$, $\alpha \neq (0,0,0)$ we get:

$$H((\alpha,0) \cdot G[s_1,s_2,s_3]) = 4$$

Now, by the construction of the last row we have that if $(\alpha_1, \alpha_2, \alpha_3) = (s_1, s_2, s_3)$ then

$$H((\alpha_1, \alpha_2, \alpha_3, 1) \cdot G[s_1, s_2, s_3]) =$$

$$H((s_1, s_2, s_3) \cdot B + (s_1, s_2, s_3) \cdot B + (1,1,1,1,1,1,1,1)) = 8$$

While for any $(\alpha_1, \alpha_2, \alpha_3) \in \{0,1\}^3$, $(\alpha_1, \alpha_2, \alpha_3) \neq (s_1, s_2, s_3)$ we get

$$H((\alpha_1, \alpha_2, \alpha_3, 1) \cdot G[s_1, s_2, s_3]) =$$

$$H((\alpha_1 + s_1, \alpha_2 + s_2, \alpha_3 + s_3) \cdot B + (1,1,1,1,1,1,1,1))$$

using the fact that $(\alpha_1 + s_1, \alpha_2 + s_2, \alpha_3 + s_3) \neq (0,0,0)$, and Equation (2), we get that this Hamming weight is exactly 4, and we are done.

4 A hard gap for $k$-COLORABILITY at gap-location 1

Consider the problem of finding a $k$-coloring of a given graph $G$, such that as many edges as possible are adjacent to two vertices of different colors.

Definition 4.1 (A consistent edge): Consider a graph $G(V,E)$ and a coloring of its vertices $\sigma : V \rightarrow \{1,2,\ldots,k\}$. we say that an edge $e = (v_i,v_j)$ is consistent regarding $\sigma$, if $\sigma(v_i) \neq \sigma(v_j)$.

Definition 4.2 The problem MAX $k$-COLORABILITY:

Input: A graph $G(V,E)$.

Problem: Find the maximum, over all $k$-coloring of the vertices in $G$, of the number of consistent edges in $G$.

For $k \geq 3$, it is NP-Hard to tell whether a graph is $k$-colorable or not. We show that for any $k \geq 3$, there exists a constant $\varepsilon_k > 0$ such that unless P=NP, there is no polynomial time algorithm which can determine whether an input graph $G(V,E)$ is $k$-colorable, or whether any $k$-coloring of $G$ has at most $(1 - \varepsilon_k)|E|$ consistent edges.
Theorem 4.3 For any \( k \geq 3 \), \( \text{MAX } k\text{-COLORABILITY} \) possesses a hard gap at location 1.

Proof: We use a reduction from \( \text{MAX } 3\text{SAT-B} \) to \( \text{MAX } 3\text{-COLORABILITY} \). Next, we can use techniques from \([\text{PY}91]\) to further reduce \( \text{MAX } 3\text{-COLORABILITY} \) to \( \text{MAX } k\text{-COLORABILITY} \) for any \( k > 3 \). In the reduction, we use a bipartite expander, which helps us to preserve the hard gap. The use of expanders in preserving gaps was first noticed in \([\text{PY}91]\). A bipartite graph on \( 2 \times n \) nodes is called a bipartite expander with degree \( d \) and expansion factor \( 1 + \gamma \), if every subset \( S \) of at most \( \frac{\gamma}{2} \) nodes of one side of the graph, is adjacent to at least \( (1 + \gamma)|S| \) nodes on the other side. Bipartite expanders on \( 2 \times n \) nodes can be efficiently constructed for any \( n \in \mathbb{N} \) \([\text{Mar}73, \text{GG}81, \text{AJ}87]\). Let us first show the reduction, and then show that if the instance \( \varphi \) of \( \text{MAX } 3\text{SAT-B} \) is satisfiable, then the output of the reduction, \( G^\varphi(V, E) \), is 3-colorable (Lemma 4.4 bellow), while if any assignment to \( \varphi \) satisfies at most a fraction of \( 1 - \epsilon \) of the clauses in \( \varphi \), then any 3-coloring of \( G(V, E) \) induces at least 
\[
\epsilon \frac{7}{2B} m \geq c|E^\varphi|
\]
inconsistent edges for some constant \( c > 0 \) (Lemma 4.5 bellow).

The reduction

We are given an instance of \( \text{MAX } 3\text{SAT-B} \), i.e., a 3-CNF formula \( \varphi \) with \( n \) variables and \( m \) clauses, such that each variable appears at most \( B \) times in the formula \( \varphi \). We use an extension of the standard reduction from 3-SAT to 3-COLORABILITY \([\text{St}73, \text{GJS}76]\). Let us shortly describe the original reduction. This reduction uses a gadget (see Fig. 1) with nine vertices and 10 edges.

![Fig. 1: The Gadget](image)

We call the top vertex \( (g_4) \) the gadget head, and the three bottom vertices \( (g_1, g_2, g_3) \) the gadget legs. The other vertices in the gadget are called the gadget body. The useful property of this gadget is that if the three legs have the same color, then any (consistent) 3-coloring of the gadget assigns the gadget head the same color too, while if the three legs do not have the same color then, for any color assigned to the head, we can complete the coloring of the body consistently.

The reduction outputs \( 2n \) vertices (the literals vertices) labeled \( x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n \), two vertices named GROUND and B, and \( m \) gadgets, one for each clause. The edges of the output graph connect \( x_i - \overline{x}_i, x_i - \text{Ground}, \) and \( \overline{x}_i - \text{Ground}, \) for \( 1 \leq i \leq n, \) it connects all the gadget heads to B, and B to GROUND. Last, it identifies the three legs of the gadget of \( C_i \) with the vertices that correspond to the literals of \( C_i \).

Our extension is the following. We duplicate the vertex B \( m \) times to get \( B_1, B_2, \ldots, B_m \), which are all connected to GROUND, and we connect them to the \( m \) gadget heads using a bipartite expander. Namely, one side of the expander (which we call the upper side) contains \( B_1, B_2, \ldots, B_m \), and the other (the lower side) contains the gadget heads. The resulting graph is best illustrated in Fig. 2.
Note that, for simplicity, we have drawn the vertex GROUND twice in the figure. Also, it is important to note that the number of edges in the output graph is $O(m)$.

**Fig. 2: The reduction from 3-SAT to MAX-3-COL**

**Lemma 4.4** If $\varphi$ is satisfiable then $G^\varphi$ is 3-colorable.

**Proof:** We use the colors T,F,G. Color the vertex GROUND with the color G. Color each literal-vertex by T if the corresponding literal is assigned TRUE by the satisfying assignment ($\tau$) of $\varphi$, or otherwise by F. Color $B_1, B_2, \ldots, B_m$ with F, and the gadget heads with T. It remains to color the body vertices of the clauses-gadgets. Recall that each gadget head is colored T and that since the truth assignment $\tau$ satisfies $\varphi$, then at least one gadget leg must be colored by T (This is the literal-vertex that corresponds to the literal that is assigned TRUE by $\tau$). By the property of the gadget we get that it is possible to complete the coloring of all the gadget-bodies vertices (consistently).

**Lemma 4.5** If there exists a 3-coloring of $G^\varphi$ which induces $\delta m$ inconsistent edges, then there is a satisfying assignment to $\varphi$ that satisfies at least $(1 - \frac{2\delta}{\gamma})m$ clauses in $\varphi$, where $1 + \gamma$ is the expansion rate of the polynomial time constructible expander that we use.

**Proof:** The lemma is trivially valid for $\delta \geq 1/4$, since for any formula there exists an assignment that satisfies at least half of its clauses. Therefore, we restrict ourselves to $\delta < 1/4$. Given a 3-coloring of $G^\varphi$, we first select names to the 3 colors, and define an
assignment to the variables of \( \varphi \). Denote the color of the vertex GROUND by \( G \). The majority color in the heads of all gadgets is denoted \( T \) (if there are two candidate colors, appearing the same number of times, select one of them arbitrarily). Note that the color \( T \) is different from \( G \) since all the \( m \) gadget-heads are connected to the ground vertex and we have less than \( m/4 \) inconsistent edges. The third color is denoted \( F \). Now, to fix the assignment for the variable \( x_i \), \( 1 \leq i \leq n \), we consider the vertex labeled \( x_i \). If it is colored \( T \), we assign the value TRUE to \( x_i \). Otherwise, the assignment to \( x_i \) is set to FALSE. We claim that this assignment satisfies at least \( (1 - \frac{2B\delta}{\gamma})m \) clauses.

By the property of the gadget, there is no (consistent) 3-coloring of the vertices of the gadget such that the head of the gadget is colored \( T \), and its three bottom vertices (the three literal vertices) are colored \( F \). Therefore, if the gadget of the clause \( C_j \), \( 1 \leq j \leq m \), has all its edges consistent, and its head colored \( T \), then one of its literals must be colored \( T \) or \( G \). In other words, if a clause \( C_j = (l_{j1} \lor l_{j2} \lor l_{j3}) \), \( 1 \leq j \leq m \), is not satisfied by our assignment, i.e., all its literals are assigned FALSE by our assignment, then one of the following conditions must be met:

1. One of the edges in the gadget of \( C_j \) is inconsistent.

2. The gadget head is not colored \( T \).

3. One of the literals in the clause is assigned FALSE by our assignment, and its vertex is not colored \( F \).

We show that these three conditions cannot be met "too many times" in the graph \( G^\varphi \) by showing that fulfillment of these conditions implies inconsistent edges. Denote by \( K_1 \), \( K_2 \) and \( K_3 \) the number of times that conditions 1, 2 and 3 are met in the graph \( G^\varphi \). Clearly, the number of clauses that are not satisfied by our assignment is at most \( K_1 + K_2 + K_3 \). We claim that

\[
K_1 + \gamma K_2 + \frac{1}{B}K_3 < \delta m
\]

Assuming this (the proof follows) we get that

\[
K_1 + K_2 + K_3 \leq \frac{2B}{\gamma} \delta m
\]

and therefore, the number of clauses that are not satisfied by our assignment is at most \( \frac{2B\delta}{\gamma} m \) as needed.

It remains to prove Equation (3), i.e., show that there are at least \( K_1 + \gamma K_2 + \frac{1}{B}K_3 \) inconsistent edges in \( G^\varphi \). We partition the edges in the graph to three disjoint subsets, and give a lower bound on the number of inconsistent edges in each subset (given \( K_1 \), \( K_2 \) and \( K_3 \)). Since the subsets are disjoint, we can sum these lower bounds into a single lower bound on the number of inconsistent edges in the graph. The first subset, consists of the edges inside the gadgets. By the definition of \( K_1 \), we have at least \( K_1 \) inconsistent edges in this subset.

Next, we consider all the edges that connect the literal-vertices between themselves and to the vertex GROUND. Recall that each literal appears in at most \( B \) clauses, and therefore, if we have \( K_3 \) clauses connected to literals, having the property of condition 3, then there are at least \( \frac{1}{B}K_3 \) literals that are assigned FALSE by our assignment, and their vertices are not
colored F. We would like to show that for each such literal, there is a unique inconsistent edge. If the vertex \( l_i \) or the vertex \( \bar{l}_i \) is colored G then there exists an inconsistent edge between that vertex and the vertex GROUND, and we are done for that literal. So, assume this is not the case. Since the vertex \( l_i \) is not colored F and not colored G, then it is colored T, and since we assigned the literal \( l_i \) with FALSE, then the vertex \( \bar{l}_i \) must be colored T also, and we get an inconsistent edge \((l_i, \bar{l}_i)\). Note that for each vertex that satisfies the above condition we have a different inconsistent edge. Therefore, we have at least \( \frac{1}{2} K_3 \) inconsistent edges in the second subset.

The remaining edges, consist of the edges of the expander, and the edges that connect the expander to the vertex GROUND. We claim that if \( K_2 \) vertices of the lower side of the expander are not colored T, then at least \( \frac{1}{2} \gamma K_3 \) edges in this set are inconsistent. Denote by \( l_1, l_2 \) and \( l_3 \) the number of vertices in the lower side of the expander that are colored T,F and G correspondingly. Recall, that \( K_2 = l_2 + l_3 \) is the number of expander lower vertices (which are gadget-head vertices) that are not colored T. We show that the number of inconsistent edges in this last set of edges, is at least

\[
\max(l_3, \gamma l_2) \geq \frac{\gamma}{2} K_3
\]

Clearly, there are at least \( l_3 \) inconsistent edges in this set, because there are \( l_3 \) different edges that connect the vertex GROUND, which is colored G, to gadget-head vertices with the same color. On the other hand, consider the \( l_2 \) expander lower vertices that are colored F. By definition of the color names, \( l_2 \) is smaller than half the number of the vertices in the lower side of the expander. Using the expansion property of the expander, these nodes have a set of \((1 + \gamma) l_2 \) neighbors in the upper side of the expander. Denote this set of neighbors by \( S \). We cannot use the expansion property again on the set \( S \), since we are not sure that it is small enough. However, we know that \( S \) is adjacent to at least \( |S| \) vertices on the lower side. Furthermore, we can associate with each vertex in \( S \), a unique neighbor on the lower side.\(^6\)

Now, only \( l_2 \) of the vertices in \( S \), can have their associated neighbors in the lower side colored F. The other \( \gamma l_2 \) members of \( S \) have their \( \gamma l_2 \) twin vertices on the lower side colored G or T. Putting it all together, we have at least \( \gamma l_2 \) vertices in \( S \) which, on one hand, are connected to a vertex colored F (by the definition of the set \( S \)), and on the other hand, are connected to unique lower vertices that are not colored F. Consider such a vertex in \( S \). If it is colored F, then we have a unique inconsistent edge between this vertex and an F-colored lower vertex. If it is colored G, then we get a unique inconsistent edge between this vertex and the vertex GROUND. The last possibility, is that this vertex is colored T. If its twin vertex is also colored T, then we have an inconsistent edge between them. Otherwise, the (unique) neighbor is colored G (since we know that it is not F), and we also end up with a unique inconsistent edge, between the (unique) lower neighbor, and the vertex GROUND.

5 The hardness of MAX NAE 3SAT and MAX 3DM

Two more problems were shown to have a hard gap at locations \(< 1 \) although it is NP-Hard to decide if an instance has its best solution at location 1. These are MAX NOT-ALL-EQUAL 3SAT [PY91] and MAX 3-DIMENSIONAL MATCHING [Kan91]. In this section

\(^6\)This is true in all known expander constructions. Yet, this property can be simply achieved (without foiling the expander) by adding \( m \) edges that connect each vertex \( i \) on one side to vertex \( i \) on the other side.
we show that these problems have a hard gap at the gap location 1. Let us begin with defining the problems.

**Definition 5.1** The problem MAX NOT-ALL-EQUAL 3SAT:
Input: A 3-CNF formula \( \varphi \)
Problem: Find the maximum, over all truth assignments to the variables of \( \varphi \), of the number of clauses that contain at least one true literal and at least one false literal.

**Definition 5.2** The problem MAX 3-DIMENSIONAL MATCHING-B:
Input: A set \( M \subseteq W \times Y \times Z \), where \( W, Y \) and \( Z \) are disjoint finite sets, and each element in \( W \cup Y \cup Z \) appears in the triplets of \( M \) at most \( B \) times.
Problem: Find the maximum, over all \( M' \subseteq M \), of the number of elements in \( W \cup Y \cup Z \) which appear exactly once in the triplets of \( M' \).

**Theorem 5.3** The problem MAX NOT-ALL-EQUAL 3SAT possesses a hard gap at location 1.

**Proof:** We first reduce 3SAT to NOT-ALL-EQUAL 4SAT, and then reduce NOT-ALL-EQUAL 4SAT to NOT-ALL-EQUAL 3SAT. In the first reduction, we simply add a new variable to all clauses. Note that for the NOT-ALL-EQUAL problem, the number of "satisfied" clauses does not change if we use an assignment \( \sigma \) or its complement \( \overline{\sigma} \) (the complement of an assignment is an assignment that gives the opposite value to each of the variables in the formula). Therefore, we can fix the new variable to always be assigned FALSE without changing the solution to this optimization problem, and the maximum number of clauses that can be satisfied in the input formula (to the reduction) is exactly the number of clauses that can have both a false and a true literal simultaneously in the output formula. In the second reduction, we treat each clause \( C_j = (l_{j1} \lor l_{j2} \lor l_{j3} \lor l_{j4}) \) that contain four literals, by adding a new variable \( y_j \), and replacing \( C_j \) with the two clauses: \((l_{j1} \lor l_{j2} \lor y_j) \land (\overline{y_j} \lor l_{j3} \lor l_{j4})\). It is easy to verify that there exists an assignment to the variables of the original formula, in which (at least) one of the literals \( l_{j1}, l_{j2}, l_{j3}, l_{j4} \) is true and (at least) one is false iff there exists an assignment to the variables of the new formula such that both sets of literals \( \{l_{j1}, l_{j2}, y_j\} \) and \( \{\overline{y_j}, l_{j3}, l_{j4}\} \) contain (at least) one true literal and (at least) one false literal. Hence, the hard gap at location 1 is preserved. Note that the width of the new hard gap is at least half the width of the original gap. This is true, because the number of clauses is at most twice the number of the original formula, and the number of clauses that can not be satisfied does not decrease. \( \blacksquare \)

**Theorem 5.4** For any \( B \geq 3 \), MAX 3-DIMENSIONAL MATCHING-B possesses a hard gap at location 1.

**Proof:** A slight modification of the original reduction from SAT to 3-DIMENSIONAL MATCHING [Kar72] works as a gap preserving reduction from MAX-SAT-B to MAX 3-DIMENSIONAL MATCHING-3. We shortly describe the reduction, following the presentation of [GJ79]. Given a formula \( \varphi \) with \( n \) variables and \( m \) clauses, in which each variable appears at most \( B \) times. We construct three disjoint sets \( W^\varphi, Y^\varphi, Z^\varphi \) and a set of triplets \( M^\varphi \subseteq W^\varphi \times Y^\varphi \times Z^\varphi \). The triplets in \( M^\varphi \) consist of \( n \) truth-setting components (one for each variable), \( m \) satisfaction-testing components (one for each clause), and a "garbage collection" mechanism.
Let \( x_i \) be a variable that appears \( d_i \) times in the formula. The truth-setting component of \( x_i \) involves “internal” elements \( a_i[k] \in W^\varphi \), \( b_i[k] \in Y^\varphi \) for \( 1 \leq k \leq d_i \), and “external” elements \( x_i[k], \overline{x}_i[k] \in Z^\varphi \) for \( 1 \leq k \leq d_i \). We call the \( a_i[k] \)’s and the \( b_i[k] \)’s internal because they appear only inside their truth-setting component. The external \( x_i[k] \)’s and \( \overline{x}_i[k] \)’s appear in their truth-setting components as well as in other components (which we describe later). The triplets making up the truth-setting component can be divided into two sets:

\[
T_i = \{ (\overline{x}_i[k], a_i[k], b_i[k]) : 1 \leq k \leq d_i \}
\]

and

\[
T_i' = \{ (x_i[k], a_i[k+1], b_i[k]) : 1 \leq k \leq d_i \} \cup \{ (x_i[d_i], a_i[1], b_i[d_i]) \}
\]

The property of this component is that any matching, that covers all internal elements \( a_i[k], b_i[k] \ 1 \leq k \leq d_i \) exactly once, contains either exactly all triplets in \( T_i \) or exactly all triplets in \( T_i' \). Thus, we get that either all elements \( x_i[k] \ 1 \leq k \leq d_i \) are covered and all elements \( \overline{x}_i[k] \) are not (we associate this with assigning FALSE to \( x_i \)), or all elements \( \overline{x}_i[k] \ 1 \leq k \leq d_i \) are covered and all elements \( x_i[k] \) are not (this is associated with assigning TRUE to \( x_i \)). Note that the number of elements produced so far is \( O(n) \) since each variable appears at most \( B \) times in \( \varphi \). (We remark that the original reduction produced \( O(nm) \) elements, having \( m \) external elements for each variable).

The satisfaction-testing component that stands for the clause \( C_j \ (1 \leq j \leq m) \) contains two internal elements \( s_1[j] \in W^\varphi \) and \( s_2[j] \in Y^\varphi \), and (at most) three external elements from the truth-setting components that correspond to the literals in \( C_j \). If \( C_j \) contains the \( k \)-th appearance of the variable \( x_i \) then we add the triplet \( (x_i[k], s_1[j], s_2[j]) \) if \( x_i \) appears positively in \( C_j \) or the triplet \( (\overline{x}_i[k], s_1[j], s_2[j]) \) if \( x_i \) appears negated in \( C_j \). These components add \( 2m \) elements to the output. So far, each internal element appears in at most three triplets of \( M^\varphi \) and each external element appears at most twice.

Note that a matching, that covers all internal elements exactly once and which does not use an external element more than once, corresponds to a truth assignment that satisfies \( \varphi \). Given an assignment \( \tau \), that satisfies \( \varphi \), we select the triplets of \( T_i \) to the matching if \( x_i \) is assigned TRUE by \( \tau \), or the triplets of \( T_i' \) otherwise. This leaves all elements \( x_i[k] \ (1 \leq k \leq d_i) \) uncovered if \( x_i \) is assigned TRUE by \( \tau \), or all elements \( \overline{x}_i[k] \) uncovered otherwise. For each clause \( C_j \), we select a literal that is assigned TRUE (such a literal must exist since \( \tau \) satisfies \( \varphi \)). The element that corresponds to the appearance of this literal in \( C_j \) is not covered, since its literal is assigned TRUE. Thus, we may choose the triplet that contains this element to cover \( s_1[j] \) and \( s_2[j] \).

In order to cover the remaining uncovered external elements, we use a garbage collection mechanism. The original mechanism is too big (it contains \( O(nm) \) elements) and does not meet the demand, that each element appears in at most three triplets. We present an apt garbage collecting mechanism later.

Consider the other direction, in which we are given a good matching and we would like to build a satisfying assignment to \( \varphi \). In order to show a hard gap in MAX 3-DIMENSIONAL MATCHING-3 we shall show that if there are “only few” violations in the given matching then there is a truth assignment that satisfies “almost all” clauses in \( \varphi \). More formally, if there exists a matching \( M' \) for which the number of internal elements which appear more than once or none at all plus the number of external elements that appear more than once is at most \( \delta m \), then there exists an assignment that satisfies more than \( (1 - \delta B)m \) clauses.
To fix an assignment to $x_i$, consider the truth-setting component of $x_i$. We assign $x_i$ with TRUE if the $M'$ contains a triplet in $T_i^l$. Otherwise, $x_i$ is assigned FALSE. Note that if this truth assignment does not satisfy a clause $C_j$ then one of the following conditions must be met.

1. The internal elements $s_1[j]$ and $s_2[j]$ do not appear in the matching $M'$.

2. $s_1[j]$ and $s_2[j]$ appear in a triplet that contains a literal which was assigned FALSE by our assignment.

Suppose condition (1) is met $K_1$ times and condition (2) is met $K_2$ times in the $M'$. The number of clauses in $\varphi$, that are not satisfied by our assignment, is at most $K_1 + K_2$. To be done, we show that number of violations in the matching $M'$ is at least $2K_1 + \frac{1}{2}K_2$.

Clearly, each time condition (1) is met, we have two unique internal elements that are not covered by $M'$. If condition (2) is met $K_2$ times in the $M'$ then there are at least $\frac{1}{2}K_2$ literals that are assigned FALSE and that have an associated external element covered by a satisfaction-testing component. We claim that the truth-setting component of this literal has either an internal element that does not appear uniquely in the matching $M'$, or an external element that appears more than once. Suppose all internal elements appear exactly once. By the property of the truth-setting component, we know that all external elements, associated with the literal that was assigned FALSE by our assignment, are covered by the truth-setting component. Thus, the external element appears in $M'$ at least twice.

It remains to show how to do garbage collection with $O(m)$ elements such that no element appears in more than three triplets. Note that the existence of a hard gap is already proven, but we must show a garbage collection mechanism in order to place the hard gap at location 1. An elegant way to solve both problems of the original garbage collection mechanism is due to Garey and Johnson (private communications). Create three independent copies of the construction described, with the rolls of $W^e$, $Y^e$, and $Z^e$ cyclically permuted between the three copies. Now, for each external element $x_i[k]$ and $\overline{x}_i[k]$ (currently included in just two triplets), add a single triplet including the three copies (one from each copy of the overall construction). This method shrinks the width of the gap by a constant factor (at location 1), and therefore is sufficient to prove the theorem. However, let us also describe an alternative garbage-collection mechanism which is less wasteful in terms of preserving the width of the hard gap.

Recall that in this stage, we have already selected triplets to the matching according to a satisfying assignment to $\varphi$. We have covered all the internal elements of the truth-setting component and of the satisfaction-testing component. We would like to cover the remaining external elements that were not covered. Consider a satisfaction-testing component that represents the clause $C_j$. This component is connected to three truth-setting components via the three elements that represent its literals. Namely, $x_{ji}[k_1]$ or $\overline{x}_{ji}[k_1]$, $x_{ji}[k_2]$ or $\overline{x}_{ji}[k_2]$, and $x_{j2}[k_3]$ or $\overline{x}_{j2}[k_3]$. Consider these three couples of element. We know that our partial matching has already covered three of these six elements (one of each couple) with the triplets of the truth-setting components. One additional element is covered by the satisfying-testing component. We do the garbage collection as follows. For each of the three couples, $x_{ji}[k_l]$ and $\overline{x}_{ji}[k_l]$ $1 \leq l \leq 3$, we construct two new internal garbage-collecting elements $g_1[ji, k_l] \in W^e$ and $g_2[ji, k_l] \in Y^e$ and we add to $M^e$ the two triplets $(g_1[ji, k_l], g_2[ji, k_l], x_{ji}[k_l])$ and $(g_1[ji, k_l], g_2[ji, k_l], \overline{x}_{ji}[k_l])$.
$g_2[j_i, k_i], \overline{x}_j[k_i])$. These triplets enable us to collect one element in each of the three couples. In this way, we may cover the elements that were not covered by the variable-setting components. However, we must leave one of these elements free for the satisfaction-testing component. Therefore, for each clause $C_j$, we add a single element $g_3[j] \in Z^*$ which is connected to the three couples of internal garbage collecting elements of the clause. Namely, we add the triplets $(g_1[j_i, k_i], g_2[j_i, k_i], g_3[j])$ for $1 \leq l \leq 3$. We use this element to cover the couple $g_1[j_i, k_i], g_2[j_i, k_i]$ that was left uncovered because its two adjacent elements (i.e., $x_j[k_i]$ and $\overline{x}_j[k_i]$) were both covered, one by a truth-setting component and the other by a satisfaction-testing component. □

6 Some more hard problems

In the spirit of "approximating the quality of the best solution", we present definitions and hardness results to the approximation versions of VERTEX COVER, SET SPLITTING, EDGE COLORING (CHROMATIC INDEX) and restricted (but still NP-Complete) versions of FEEDBACK VERTEX SET and FEEDBACK ARC SET.

Let us start with the definitions. We say that an edge $(v_i, v_j)$ in a graph $G(V, E)$ is covered by a subset $V' \subseteq V$ if $v_i \in V'$ or $v_j \in V'$.

**Definition 6.1** The problem MAX VERTEX COVER-B:

*Input:* A graph $G(V, E)$ with degree at most $B$, and an integer $K$.

*Problem:* Find the maximum, over all subsets $V' \subseteq V$ with cardinality $K$, of the number of edges in $G$ that $V'$ covers.

Note the difference between the optimization problem MIN VERTEX COVER, as described in [PY91]. The problem there, was to minimize the size of the vertex cover (which covers all edges). In MAX VERTEX COVER, we are given the size of the cover, $K$, in the input, and we are trying to maximize the number of edges that are covered.

**Definition 6.2** The problem MAX SET SPLITTING:

*Input:* A collection $C$ of subsets of a finite set $S$.

*Problem:* Find the maximum, over all partitions of $S$ into two subsets $S_1$ and $S_2$, of the number of subsets in $C$ that are not entirely contained in either $S_1$ or $S_2$.

We say that a vertex $v \in V$ is consistent regarding an edge-coloring of a graph $G(V, E)$, if no two edges of the same color are adjacent to $v$.

**Definition 6.3** The problem MAX $k$-EDGE COLORABILITY:

*Input:* A graph $G(V, E)$ of degree $k$.

*Problem:* Find the maximum, over all edge-colorings of $G$ with $k$ colors, of the number of consistent vertices.

**Definition 6.4** The problem MAX sqrt-FEEDBACK VERTEX SET:

*Input:* A directed graph $G(V, E)$, and an integer $K$.

*Problem:* Find the maximum, over all $V' \subseteq V$ with $|V'| = K$, of the number of simple directed cycles of length $\leq \sqrt{|V|}$ in $G$ that contain a vertex of $V'$.
Definition 6.5 The problem MAX sqrt-FEEDBACK ARC SET:
Input: A directed graph $G(V, E)$, and an integer $K$.
Problem: Find the maximum, over all $E' \subseteq E$ with $|E'| = K$, of the number of simple
directed cycles of length $\leq \sqrt{|V|}$ in $G$ that contain an edge of $E'$.

Theorem 6.6 The following problems possess a hard gap at location 1:

1. MAX VERTEX COVER-B
2. MAX EDGE COLORING (CHROMATIC INDEX)
3. MAX SET SPLITTING
4. MAX sqrt-FEEDBACK VERTEX SET and
5. MAX sqrt-FEEDBACK ARC SET.

Sketch of proof:
(1) We use the identity reduction from MIN VERTEX COVER-B (a problem that was shown hard in [PY91]). If there is a vertex cover of $G$, of cardinality $K$, that covers $\geq \left( 1 - \epsilon \right) |E|$ edges in $G$, then there is a cover of all edges with $\leq K + \epsilon |E|$ vertices. Note that $|E| \leq \frac{K}{2} |V|$, and recall that the hard gap for MIN VERTEX COVER-B was shown hard for $K = \Theta\left( \sqrt{|V|} \right)$.
(2) Use the original reduction of MAX-SAT to k-EDGE COLORABILITY (see [Hoy81, LG83]), only let the domain of the reduction be MAX-3SAT-B (shown hard in [PY91]). Clearly, if there is a satisfying assignment to the input formula, then there is an edge-coloring of the output graph with $k$ colors, such that all the vertices are consistent. It is left to show that if there is a $k$-coloring of the edges with a small number of inconsistent vertices, then there is an assignment that satisfies almost all clauses in the input formula. This can be done using a similar book-keeping to the one of Lemma 4.5.
(3) The trivial reduction from MAX NOT-ALL-EQUAL 3SAT works.
(4) We reduce MAX VERTEX COVER to MAX sqrt-FEEDBACK VERTEX SET by replacing each edge $(v_i, v_j)$ in the input graph $G(V, E)$ with two directed paths of length $\beta$. One of these directed paths, leads from $v_i$ to $v_j$ and the other $v_j$ to $v_i$. (Recall that the original reduction in [Kar72], uses a single directed edge instead of each path). Let $G'(V', E')$ be the resulting graph. $\beta$ is chosen such that $2\beta < \sqrt{|V'|}$ and $3\beta > \sqrt{|V'|}$. We fix the number of vertices in the feedback set, $K$, to be the same as the number $K$ in the input problem. To see that the reduction is valid, note that the shortest cycle in $G'$ is of size $2\beta$, and each such cycle corresponds to a unique edge in $G$. By the definition of $\beta$, the length of this cycle is less than $\sqrt{|V'|}$. All cycle in $G'$ that do not origin from a single edge in $G$, are of size $\geq 3\beta$ which is bigger than $\sqrt{|V'|}$ and therefore are not relevant. Hence, the number of relevant cycles in $G'$ is exactly the number of edges in $G$, and if there is a subset of $V'$ which contains a vertex in a fraction of $(1 - \epsilon)$ of the cycles in $G'$, then there is a subset of $V$ that covers (at least) a fraction of $(1 - \epsilon)$ of the edges in $G$.
(5) Following [Kar72], we use an extension of the previous reduction (from MAX VERTEX COVER). First, we duplicate each vertex $v$ in the input graph $G(V, E)$, to $< v, 1 >$ and $< v, 2 >$, and we connect these vertices by a directed edge from $< v, 1 >$ to $< v, 2 >$ (for each $v \in V$). For each original edge $(u, v) \in E$, we construct two directed paths of length
\( \beta - 1 \). One from \(< u, 2 >\) to \(< v, 1 >\), and the other from \(< v, 2 >\) to \(< u, 1 >\). These paths are, again, composed of new vertices, that are unique to their paths. The number \( \beta \) satisfies 

\[ 2\beta < \sqrt{|V|} \text{ and } 3\beta > \sqrt{|V|} \]

Note that any cycle that is connected to \(< v, 1 >\) must traverse the edge \(< v, 1 >, < v, 2 >\). The rest of the proof is practically the same.  

Remark 6.7 The approximation of MAX SET SPLITTING remains hard even if all subsets in \( C \) are of cardinality \( \leq 3 \) (see the proof sketch).

7 Acknowledgements

I am greatly indebted to Oded Goldreich for many insightful discussions, without him this work would have not been possible. I would like to thank Benny Chor and Amos Beimel, whose suggestion of the problems in Coding Theory, initiated this work. I am grateful to David S. Johnson for his quick and profound answer to a query I made. And last, I am grateful to Amos Beimel and Seffi Naor for helpful discussions.

References

A : The Hardness of Satisfiability

In this appendix, we define two satisfiability problems which are referred frequently during this work. We also state the hardness of approximating these problems.

Definition A.1 The problem MAX-3SAT:
Input: A 3CNF formula with \( n \) variables and \( m \) clauses.
Problem: Find the maximum, over all truth assignments to the variables of \( \varphi \), of the number of clauses in \( \varphi \) that are satisfied.

Definition A.2 The problem MAX-3SAT-B:
Input: A 3-CNF formula \( \varphi \) with \( n \) variables and \( m \) clauses, where any variable \( x_i, 1 \leq i \leq n \), appears in \( \varphi \) at most \( B \) times.
Problem: Find the maximum, over all possible truth assignments to the variables of \( \varphi \), of the number of clauses that can be satisfied.

References:


Note that an input to this problem satisfies $n^2 = \text{constant}$ (in fact $\frac{n}{3} \leq m \leq \frac{Bm}{3}$).

The following theorem is the foundation of all lower bounds in this work.

**Theorem A.3 [ALMSS92, PY91]:** For any $B \geq 4$, there exists a constant $\epsilon > 0$, such that unless $P=NP$, there is no polynomial algorithm that distinguishes between instances of MAX-SAT-B which are satisfiable, and instances (formulae) that any assignment to their variables satisfies at most a fraction of $1 - \epsilon$ of their clauses.