# The intrinsic hardness of Structure Approximation

by

©Md Renesa Nizamee

A Thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

**M.Sc in Computer Science**

**Department of Computer Science**

Memorial University of Newfoundland

**Month and Year Submitted**

St. John's Newfoundland

# Abstract

A natural question when dealing with problem that are NP hard to solve is whether it's possible to compute a solution which is close enough to the optimal solution for practical purposes. The usual notion of "closeness" is that the value of the solution is not far from the optimal value. In this M.Sc thesis, we will focus on the generalization of this notion where closeness is defined with respect to a given distance function. This framework, named "Structure approximation", was introduced in 2007 paper by Hamilton, Müller , van Rooij and Wareham, who posed the question of complexity of solving NP-hard optimization problems in this setting.

Here, we are going to examine the complexity of structure approximation and show that at least for some choices for distance function any non trivial structure approximation is as hard to compute as the original problem. In particular we will show the most natural distance function in a computational complexity setting treated as binary strings which is Hamming distance. We will survey the relevant results from other related problems and show how they relate to our problem. We will also survey the techniques that can be used to solve this problem to prove inapproximability results such as Error correcting codes.

# Acknowledgements

Text for the acknowledgements.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

A problem is in $P$ means it has a polynomial time algorithm and in $NP$ means though it might be hard to find a solution but once found it can be verified in polynomial time. P versus NP problem is a major unsolved problem in theoretical computer science. The Clay Mathematics Institute in Cambridge, MA, has named P versus NP as one of its Millennium problems, and offers \$1 million to anyone who provides a verified proof. Lots of important problems are $NP$ problems and we would love to solve them in polynomial time. Suppose an advertising company on Facebook is wondering what is the biggest group of people who are friends with each other. This question is an instance of a well known NP hard problem called Clique. Another beautiful problem is called Knapsack where we are given with a set of items and we have to choose the most profitable ones within a certain bound. This problem naturally occurs in transportation. Sending a spaceship to space needs a calculated decision of which item to take within a certain weight and size bound. These problems are hard to solve being $NP$ hard problems. Though we face these kinds of problems in real life

all the time but being $NP$ hard problems those are hard to solve. The best we can do is rather than finding the exact solution we can look for a solution that is close to the optimal solution.
What does it mean to find a solution close to the optimal one? There are many ways of looking at it. The standard way is to when we want to find something that would be almost as good as optimal solution such as, when we are looking for a Maximum clique in a graph we may be content with a clique that has a fewer less node than the maximum. Standard notion of approximation is we try to get a solution which is as good as some value function as an optimal solution. But sometimes this real approximation algorithms are not exactly what we want.
One of the motivations from the paper [HMvRW07] was that in Cognitive psychology, people talk about belief systems , they really want certain beliefs to be the same as the actual belief . From another paper [SY03] there was a beautiful example of the structure seen on a tomogram. A radiograph (x-ray) of a selected layer of the body made by tomography. A tomogram is a two-dimensional image representing a slice or section through a three-dimensional object. So, we are not getting the correct structure of the object in the x-ray film through a tomogram. Thoguh given many x-ray images, the internal structure can be determined but it is $NP$ hard to compute. But we are not interested in just something that resembles superficially over there, we want to know what the structure of the solution is. This kind of approximation Hamilton, van Rooij, Miller and Wareham [HMvRW07] called it Structure approximation where rather than focusing on the value of the solution, what we really want to know is the structure of the solution. That's why they define structure approximation problem in which there is an additional parameter called the distance function. Rather than comparing the values of two solutions they are looking at the distance of two solutions and trying to optimize that. Here, we will mostly use Hamming distance

as the distance function as we will be dealing with binary strings.

## 1.2 Chapter Organization

In the following chapters we will talk about various approaches to approximation in chapter 2, Inapproximibility results for different problems in chapter 3 where we will find out different distance functions used to prove these results. Then we will talk about some positive results on structure approximation in chapter 4 followed by conclusion in chapter 5.

# Chapter 2

# Various approaches to approximation

—-

## 2.1 Computational complexity setting

There are many important computational problems that are tough to solve in a best possible way. As a matter of fact, most of those problems are NP-hard, that means no polynomial-time algorithm exists that resolves the problem in a most favourable way unless $P = NP$.

**Definition 1.** *The class $P$ of polynomial-time decidable languages is* $P = \bigcup_{k\geq 1} Time(n^k)$.

For some function $f$ :$\mathbb{N} \rightarrow \mathbb{N}$, Time($f(n)$) = $\{L|L$ is decided by some TM in at most $f(n)$ steps$\}$.

$P$ is known to contain many natural problems, including the decision versions of linear programming, calculating the greatest common divisor, and finding a maximum matching.

**Definition 2.** *Let $L \subseteq \Sigma^*$. We say $L \in NP$ if there is a two-place predicate $R$ $subseteq \Sigma^* \times \Sigma^*$ such that $R$ is computable in polynomial time, and such that for some $c, d \in \mathbb{N}$ we have for all $x \in \Sigma^*$, $x \in L \Leftrightarrow \exists y \in \Sigma^*$, $|y| \leq c|x|^d$ and $R(x, y)$.*

**Definition 3.** *[AB09] We say a language $A \subseteq 0, 1^*$ is a polynomial-time Karp reducible to a language $B \subseteq 0, 1^*$ denoted by $A \leq_p B$ if there is a polynomial-time computable function $f : 0, 1^* \to 0, 1^*$ such as that for every $x \in 0, 1^*$, $x \in A$ if and only if $f(x) \in B$. We say that $B$ is $NP - hard$ if $A \leq_p B$ for every $A \in NP$. We say that $B$ is $NP - complete$ if $B$ is $NP - hard$ and $B \in NP$.*

It is easy to see that $P \subseteq NP$, since any problem that can be solved in polynomial time can also have its solutions verified as correct in polynomial time. [CLRS01]. Maximum Clique problem (Max-Clique) could be a common example of NP-hard problem: Given a group of vertices some of which have edges in between them, the maximum clique is the largest subset of vertices in which each point is directly connected to every other vertex in the subset.

**Definition 4.** *In a decision problem, given an input $x \in 0, 1^*$, we are required to give a YES/NO answer.*

**Definition 5.** *In a search problem, given an input $x \in 0, 1^*$ we want to compute some answer $y \in 0, 1^*$ that is in some relation to $x$, if such a $y$ exists.*

## 2.2 Heuristic vs. Approximation algorithm

When we want to solve a problem in real life, we are interested in finding a solution, not just knowing if one exists. Yet we have always formulated problems in terms of decision, meaning we pose a question and want to know whether the answer is yes or no. In fact, for every problem we have considered, there is a corresponding search problem.

Every NP problem comes in two versions, a decision version and a search version. For example: Problem: SAT Input: $<\Phi>$ where $\Phi$ is a CNF formula Decision Problem: Is $\Phi$ satisfiable? Search Problem: Find a satisfying assignment to $\Phi$ if one exists, else output $\tau$. Here $\tau$ is a special symbol used to indicate that there is no solution.
The Sat problem we have looked at, and shown to be NP-complete, is the decision problem. In the search version of the problem, we can't stop merely at saying "'yes"' or "'no"' to the question of whether the given formula is satisfiable; if the answer is "'yes"' we must actually find and output a satisfying assignmen

**Definition 6.** *An Heuristic model is a computational method that uses trial and error methods to approximate a solution for computationally difficult problems. It does not aim to find the optimal solution, sacrificing optimality for improved runtime.*

As we can see from the definition that an heuristic algorithm may ensure a fast solution but it doesn't ensure the accuracy of the solution. This method can be useful for simpler problems or where we simply don't care about the correctness of the solution rather we only care about the speed at which the solution is produced.
At this instant the question that arises is what should we do when difficult problems come along, for which we don't expect to find polynomial-time algorithms or heuristic methods doesnt apply? Algorithms that have exponential running time are not effective, if the input size isn't really small. As a result, we need to look for a solution approximately since absolutely the optimal solution is not before us. Most likely we would like to have a guarantee of being close to it. As an example, we can have an algorithm for Euclidean TSP that always develops a tour whose length is at most a factor $\rho$ times the minimum length of a tour, for a small value of $\rho$. An algorithm which produces a solution which is ensured to be within some factor of the optimum is called an approximation algorithm.
As mentioned earlier, NP-hard problems has no polynomial time algorithms unless

$P = NP$ and to find out the solution of those problems the best possible way is to use Approximation algorithm which will return the solution within some factor of the optimum or exact solution. Approximation algorithms also bring down the running time of the problem which could have exponentially huge running time if Approximation Algorithms were not used.

So, the performance of Approximation algorithm is an important issue. While it is assumed that every approximation algorithm is polynomial but there exist a lot of polynomial algorithm which can be inefficient and impractical. So, in order to get better approximation bound one can invest more running time to the algorithm. From this trade-off between approximation result and running time the notion of Approximation schemes arise. [Hoc97]

PTAS and FPTAS are those approximation schemes. As we may guess that the better the approximation, the larger may be the running time. A problem has PTAS if the running time is polynomial in the size of the input and a problem has FPTAS if the running time is polynomial both in size of the input and the inverse of the performance ration($1/\epsilon$).[ACG+99]. So, the classes of problems that has FPTAS are contained in classes of problems that has PTAS. For structure approximation we have sPTAS and sFPTAS as structure approximation scheme.

**Definition 7** (PTAS)**.** *Let $P$ be an NP optimization problem. An algorithm $A$ is said to be a polynomial time approximation scheme (PTAS) for $P$ if, for any instance $x$ of $P$ and any rational value $r > 1$, $A$ when applied to input (x,r) returns an r-approximate solution of $x$ in time polynomial in $|x|$. [ACG+99]*

**Definition 8** (FPTAS)**.** *An algorithm $A$ is said to be a fully polynomial time approximation scheme (PTAS) for $P$ if, for any instance $x$ of $P$ and any rational value $r > 1$, $A$ when applied to input (x,r) returns an r-approximate solution of $x$ in time polynomial both in $|x|$ and in $1/(r-1)$. [ACG+99]*

### 2.2.1 Value-approximation algorithm

Many types of approximation algorithms have already been defined.

**Definition 9** (Value-approximation algorithm)**.** *If the $val(x, A(x))$ is within some absolute or relative factor of $optval(x)$; we refer to it as value-approximation algorithm.*

Some classes of value-approximation are defined as following:

**Definition 10** (Additive value-approximation (v-a-approx) algorithm)**.** *Given an optimization problem $\pi$ and a no-decreasing function $h : N \to N$, an algorithm $A$ is a polynomial time $h(|x|)$ additive value-approximation (v-a-approx) algorithm for $\pi$ if for every instance $x$ of $\pi$, $|optval(x) - val(x, A(x))|) \leq h(|x|)$ and $A$ runs in time polynomial in $|x|$. [HMvRW07]*

**Definition 11** (Ratio value-approximation (v-r-approx)algorithm)**.** *Given an optimization problem $\pi = (I, cansol, val, goal)$ and a non decreasing function $h : N \to N$, an algorithm $A$ is a polynomial time $h(|x|)$ ratio value-approximation (v-r-approx)algorithm for $\pi$ if for every instance $x \in I$ , $R(x, A(x)) \leq h(|x|)$, where $R(x, A(x)) = \frac{optval(x)}{val(x,A(x))}$ (if goal=MAX) or $\frac{val(x,A(x))}{optval(x)}$ (if goal=MIN), and $A$ runs in polynomial time in $|x|$. [HMvRW07]*

### 2.2.2 Positive approximation results for different problems

A question may arise that how good can be an approximation algorithm? A nice sorting of these approximation results can be found in [Hoc97] where the author sorted the results according to "good", "better", "best", "better than best", "wonderful" categories. [see table 2.1]

Table 2.1: Approximation results of different problems

| **Category** | **Category definition** | **Example** |
|---|---|---|
| Good | Has a $\delta$-approximation for some constant $\delta$ | Vertex cover problem, MAX CUT |
| Better | Has an Approximation Scheme(PTAS, FPTAS) for the problem | KNAPSACK, Minimum makespan |
| Best | Has a polynomial c-approximation algorithm | k-Center problem |
| Better than best | Has a certain asymptotic performance ratios | Bin packing, Edge coloring |
| Wonderful | Delivers solution within 1 of the optimum | Edge coloring in simple graphs, Coloring of planar graphs, Min max degree spanning tree |

## 2.3 Problem definitions

We will see the definitions of the problems that has been used frequently here.

**Definition 12** (Clique). *A clique in an undirected graph $G = (V, E)$ is a subset of the vertex set $C \subseteq V$, such that for every two vertices in $C$, there exists an edge connecting the two.*

**Definition 13** (Max-Clique). *A maximum clique is a clique of the largest possible size in a given graph*

**Definition 14** (Knapsack). *Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.*

**Definition 15** (Shortest Vector). *Given a lattice $L$, Shortest Vector problem asks to find the shortest non-zero vector in $L$ .*

Figure 2.1: Test pspicture

**Definition 16** (SAT)**.** *Given a collection $C = c_1, c_2, ..., c_m$ of clauses, where each clause consists of a set of literals (representing the disjunction of those literals) over the finite set of Boolean variables $U = u_1, u_2, ..., u_n$, is there an assignment of truth values to $U$ which makes every clause true ?*

**Definition 17** (3-SAT)**.** *A version of the SAT problem in which every clause has 3 literals.*

## 2.4 Inapproximability Results

## 2.5 Structure Approximation

When we talk about Approximation, we mainly talk about value-approximation where the value of the candidate solution is what we care about. But there can be another possibility where the structure of the solution is considered important rather than the value. This kind of approximation was discussed by Hamilton, Müller , van Rooij and Wareham in their paper "‘Approximating Solution Structure"’where they posed the question of complexity of solving NP-hard optimization problems in Structure Approximation framework.

In this paper they propose definitions of structure appoximation and it’s classes. They also try to show the relation between structure and value approximation for different problems. They also ask some interesting question regarding this framework. In this section we mainly will see a summery of their work on structure approximation.

**Definition 18** (Structure-approximation algorithm)**.** *Given an optimization problem* $\Pi$ *and a sd-function d, and a non-decresing function* $h : \mathbb{N} \to \mathbb{N}$*, an algorithm A is a polynomial-time* $h(|x|)/d$ *structure-approximation(s-approx) algorithm if for every instance x of Pi,* $d(A(x), optsol(x)) \leq h(|x|)$ *and A runs in time polynomial in* $|x|$*.*

Here, the degree of similarity of structures between candidate solution and optimal solution is measured by a distance function which they call solution distance(sd) function.

**Definition 19** (Solution distance(sd) function)**.** *Let* $d : \Sigma^* x \Sigma^* \to \mathbb{N}$ *be a solution distance(sd) function associated with an optimization problem Pi such that for an instance x of Pi and* $y, y' \in cansol(x), d(y, y')$ *is the distance between these solutions.*

A sd-function $d$ is a metric and it has four properties: 1. For all $x, d(x, x) = 0$ 2. For all distinct $x$ and $y$ , $d(x, y) > 0$ 3. For all $x$ and $y$ , $d(x, y) = d(x, y)$ (symmetry) 4. For all $x$, $y$ and $z$ , $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

For most of the examples shown in this paper the authors had used Hamming distance as the distance function. For LCS, Edit distance was used as the distance function about which we will discuss in a later chapter.

[Hamming distance] The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

# Chapter 3

# Lower Bounds/ Inapproximibility results

### 3.0.1 General functions

### 3.0.2 Hamming lower bounds

—the following papers are under this section

### 3.0.3 Gal, Halevi, Lipton and Petrank: Partial and Approximate NP-witnesses

"‘Is it easier to find parts of the solution than getting it all?"’ This is the exact question which motivated Gal, Halevi, Lipton and Petrank [GHLP99] to find out the "‘Partial approximation"’ for several important problems in NP such as, 3SAT, Graph Isomorphism, the Shortest Lattice Vector problem, Graph 3-Colorability, Vertex Cover and Clique. They show in this paper that the answer to the initial question asked by them is "‘No"’ and they find out that retrieving parts of the solution is as hard as retrieving the full solution.

So, one can easily get confused with "‘finding parts of the solution"’, "‘the actual solution"’ and "‘their relationship"’. Suppose we look at the problem Graph Isomorphism. Their result [GHLP99] implies that finding a small part of the isomorphism is enough rather than build the whole isomorphism. So, if we can find a way to construct a small part of the isomorphism; with their technique we can easily build the whole isomorphism.

One more thing that should be considered with their technique is the issue in fault toralence. That means if a solution is send to us through a communication channel which ommits a part of it, can we still recover the full solution? In this paper they provide a positive answer for this problem by using "‘Omission-correcting algorithms"’ for those specific problems.

Their work is also motivated by cryptographic applications where a important question in this particular field is the relation between retrieving partial information about a secret and computing the whole secret. They provide some insight into this interesting question. They also try to find out the power of oracles that only supply part of the solutions.

### 3.0.4 Some results

1. (Satisfiability (SAT)): For any $\epsilon > 0$, given a CNF formula over $n$ variables it is possible to construct in probabilistic polynomial time another formula $\Phi^1$ variables (with $|\Phi'| = |\Phi| + n^{O(1)}$, such that with probability almost 1, given any $N^{1/2+\epsilon}$ bits from a satisfying assignment to $\Phi'$, one can efficiently construct a satisfying assignment to $\Phi$. [GHLP99]

   They try to erase upto 1/2 of the variables first but their solution doesn’t work as they failed to implement the method of amplification which [FLN00] successfully implemented later on (it will be discussed later on this chapter). After that they try to erase less than half of the variables and use padding technique to get their desired result which they succeed. Atlast they try to erase nearly all the bits(variables) and surprisingly found out that it has better correction ability of erasure codes.

2. (Graph Isomorphism) : There is a (deterministic) polynomial time procedure that on a given pair of graphs $(G, H)$ constructs adaptively $l = poly(n)$ oracle queries $(G_1, H_1), ...., (G_l, H_l)$ such that if on each query the oracle answers with a map on only $O(logn)$ of the variables (which is a part of isomorphism between the two graphs in this query), then procedure finds an isomorphism between $G$ and $H$. [GHLP99]

3. (Other NP-complete problems): In this paper they also show results for some more NP-complete problems such as the Shortest Lattice Vector problem, Graph 3-Colorability and they claim that they had also solved the results for Vertex Cover and Clique and also say that for Clique the proof is even more straight forward where they can use standard reduction without padding.

### 3.0.5 Kumar and Sivakumar: hard-to-approximate witnesses for all NP languages

Motivated by [GHLP99] as well as then-recent PCP theorem, Kumar and Sivakumar [KS99] set out to generalize the results of [GHLP99] to all of NP problems. They presented several constructions based on list-decodable codes (in fact, now their paper is most cited for their code constructions). There, [KS99] considered both the erasure setting of [GHLP99] (where only a subset of bits is known, but known bits are all correct), and a “noisy proof system” setting much more akin to the structure approximation with respect to Hamming distance: a witness was guaranteed to agree with some correct witness on at least a given fraction of bits. Their main idea can be described as follows (following the notation in [FLN00]).

**Definition 20.** *Let $L$ be a language in NP. Then by definition there is a polynomial-time computable relation $R_L(x, y)$ such that $x \in L$ iff $\exists y, |y| \leq |x|^c \wedge R_L(x, y)$, where $c$ is a constant. Now, consider a different relation $\hat{R}_L(x, w)$, with $|w| \leq |x|^d$, defined as $\hat{R}_L(x, w) \equiv \exists y, (w = Code(y) \wedge R_L(x, y))$. Here, $Code(y)$ is an error-correcting code (more on it later).*

Thus, if there is a polynomial time algorithm that decodes $w = Code(y)$ to obtain $y$, then the relation $\hat{R}_L$ is a polynomial-time relation.

Now, assume that for some specific code such a decoding algorithm exists. Also assume that this code can recover a string with $1/2 - \epsilon$ fraction of errors (possibly in the list-decoding regime). Then to decide $L$ it is enough to find a $1/2 + \epsilon$ approximation of $w$. That is, approximating $w$ to within $1/2 + \epsilon$ is as hard as solving the original problem; in particular, if $L$ is NP-complete, then such approximation is NP-hard.

Note that information-theoretically this is not an optimal witness: $w$ can be significantly larger than $y$, so half the bits of $w$ could be longer than all of $y$. This is one of the issues addressed by [FLN00].

**Theorem 1** ([KS99])**.** *Given an approximation $w_1$ to a binary string $w$, it is possible to recover $w$ if $w_1$ agrees with $w$ on at least $1/2 + \epsilon$ fraction of bits, where $\epsilon \geq 1/|w|^{1/5}$.*

Another way to state it is that there is a way to recover a witness given a string that agrees with it on at least $|w|/2 + |w|^{4/5+\epsilon'}$ bits, for $\epsilon' > 0$. The proof of this claim follows from the constructions for erasure codes in the main body of the paper, together with the Johnson's bound.

More specifically, the paper provides three constructions of erasure codes. All three constructions consist of the Reed-Solomon code as an outer code, and three different inner codes. That is, a string $y$ is first viewed as a sequence of evaluations of a polynomial $p_y(u) = \Sigma_j y_j u^j$ over $u$ $F_q$ for a chosen field $F_q$ with $q$ elements. Treating this as a binary string, each of the $q$ terms is $\log q$ bits in length; these $\log q$ bit blocks are each encoded by an "inner code". The choice of the inner code determines the final parameters.

**Theorem 2** ([KS99])**.** *For every language $L \in NP$, every witness predicate $R_L(x, y)$, and every $\epsilon > 0$ there exists a witness predicate $\hat{R}_L(x, w)$ as defined above that given $N^\delta$ bits of $w$, recovers $y$ satisfying $R_L(x, y)$ in polynomial time. The three constructions give the following parameters:*

1. *(Using Hadamard inner code): $\delta = 3/4 + \epsilon$ and $|w| = q^2$, where $q$, the field size, is a power of two $\geq (4n)^{1/3\epsilon}$.*

2. *(Using a probabilistic construction for the inner code): $\delta = 1/2 + \epsilon$, and $|w| = q^{1+\alpha}$, for $\frac{1/2-\epsilon}{1/2+\epsilon} < \alpha < \frac{1/2+\epsilon}{1/2-\epsilon}$. However, to construct such an $\hat{R}_L$ an algorithm with a slightly superpolynomial running time $(O(2^{(\log q)^3})$ is needed.*

3. *(Using $l$-wise $\delta$-biased sample space) $\delta = 2/3 + \epsilon$, and $|w| = q^{1+\alpha}$, for $\frac{1/3-\epsilon}{2/3+\epsilon} < \alpha < \frac{1/6+\epsilon+\gamma}{1/3-\epsilon}$, where $0 < \gamma < \epsilon$ is a very small constant.*

Those are the parameters on which [FLN00] is based. We omit the technical code-theoretic details of the constructions, since we are using the codes in the black-box fashion.

Relatively recently a better construction of such codes appeared due to Guruswami and Rudra [GR08], (journal version [GR11]). They produce explicit codes list-decodable up to $1/2 + \epsilon$ fraction of errors. Moreover, their codes produce $|w| = O(n^3/\epsilon^{3+\gamma})$ for a small constant $\gamma$ (or $\tilde{O}(n/\epsilon^{5+\gamma})$; note that the optimal non-constructive bounds are $O(n/\epsilon^2)$. Thus, where the codes of Kumar and Sivakumar can recover a witness string

$w$ from $|w|/2 + |w|^{4/5+\epsilon'}$ bits, for $\epsilon' > 0$, codes of Guruswami and Rudra can recover $w$ given $|w|/2 + |w|^{2/3+\epsilon'}$ bits.

So what is the relevance of Kumar and Sivakumar work to the structure approximation setting? Intuitively, their result (as improved by Guruswami and Rudra) states that for every language in NP, there exists a witness predicate, with respect to which it is as hard to structure-approximate a witness $w$ to within Hamming distance $|w|/2 + |w|^{2/3+\epsilon'}$ as it is so solve the original problem. Thus, for an NP-complete $L$, such witness is NP-hard to approximate much better than by taking a random string (note that a random string with high probability agrees with a witness on $|w|/2 + |w|^{1/2}$ bits. ) And therefore for such a predicate, with distance function being the Hamming distance, the "structure approximation by dumb luck" of [HMvRW07] is the best possible.

### 3.0.6 Sheldon and Young: Hamming Approximation of NP Witnesses

Guruswami and Rudra briefly mention an unpublished manuscript of Sheldon and Young [SY03] that achieves hardness of structure approximation for SAT $|w|/2+|w|^{\eta}$ for any $\eta$. They were motivated by Feige, Langberg and Nissim's work [FLN00] and set to strengthen their hardness result. They show, for the universal NP-complete language, no deterministic polynomial-time algorithm can achieve Hamming distance $n/2$ $+O(\sqrt{nlogn})$ (unless P=NP). Also, no randomized polynomial-time algorithm can achieve $n/2$ $+o(\sqrt{nlogn})$ with probability $1 - 1/n^{O}(1)$(unless RP=NP)[SY03]. Another contribution of Sheldon and Young's paper was to find out the lower bounds for randomized algorithms. This result is shown in the following theorem.

**Theorem 3.** *Fix any $\epsilon > 0$. For the universal NP-complete language $\nu$ and it's* *witness relation $R_\nu$, no polynomial-time algorithm achieves Hamming distance $n/2 + \sqrt{\epsilon n \ln n}$ with probability $1 - O((n^(2\epsilon + 1) sqrt \epsilon \ln n)^-1)$.*

Here we will discuss their result for SAT.

**Lemma 3.0.1.** *Suppose that, for SAT (with the natural witness relation), there exists $\epsilon > 0$ such that some polynomial-time algorithm $A$ achieves Hamming distance $n/2 - n^\epsilon$. Then P=NP.*

*Proof.* They prove the lemma by describing a polynomial time algorithm $A'$ that solves a SAT in polynomial time. Given a SAT instance $I$ it create new instance $I'$ from $I$ by duplicating a variable $X$; $m$ times where $m = \lceil 1 + (n/2)^1/\epsilon - n \rceil$. That means $A'$ adds to $I$ new variables $X^1, X^2, ...X^m$ and adds new clauses $(X^1 = X^2) \wedge (X^2 = X^3) \wedge ... \wedge (X^(m-1) = X^m)$.
Algorithm $A'$ runs on the instance $I'$ of $A$ to achieve hamming distance $(m+n)/2 - (m+n)^\epsilon$. We assume $b$ to be a true or false value which is assigned to atleast $m/2+1$ copies of the variable $X$ in $I'$. We set $X = b$ and substitue $b$ for $X$ in $I'$ and simplify the resulting formula to get a new formula $I''$ and use recursion on $I''$ thus assigning values to all the variables remaining. [SY03] $\square$

Suppose we have a SAT formula, $(x \vee y) \wedge (\bar{x} \vee \bar{z}) \wedge (\bar{y} \vee \bar{u})$ and we assume $\epsilon = 1/3$. It has 4 variables in the form of x,y,z and u so, $n$ will be 4 here. As we know from the lemma $m = \lceil 1 + (n/2)^1/\epsilon - n \rceil$ so, $m = \lceil 1 + (4/2)^3 - 4 \rceil = 5$ here.

## 3.1 Main tool: Error correcting codes

**Definition 21.** *[FLN00]* Error correcting code-ECC *Let $\Sigma$ be an alphabet of size $q$. An $[n, k, d]_q$ error correcting code is a function $C : \Sigma^k \to \Sigma^n$ with the property that for every $a, b \in \Sigma^k$ we have a $dist(C(a), C(b))$ is greater than or equal to $d$.*

**Definition 22.** *[FLN00]* List decodable ECC *Let $\Sigma$ be an alphabet of size $q$. An $[n, k, d]_q$ error correcting code $C$ is $\delta$ list decodable if there exists a Turing Machine $D$(the list decoder) which on input $c \in \Sigma^k$ outputs in poly(n)time a list containing all words $a \in \Sigma^k$ that satisfy $dist(C(a), C(b)) \leq \delta$.*

**Definition 23.** Erasure Code

**Definition 24.** *[Sud96]* Reed Solomon Code *For a finite field $F$ of size $n$ and parameter $d$, the Reed Solomon Code is an $[n, d+1, n-d]$-code over $F$, whose codewords are the strings $(p(0), p(w), p(w^2), ...p(w^{|F|-1}))_p$, where $w$ is some fixed primitive element of $F$ and $p$ ranges all polynomials of degree at most $d$ over $F$.*

**Definition 25.** *[Sud00]* Hadamard Code *For any $k$, the Hadamard code $H_k$ is a $(n = q^k, k)_q$ code with distance $(1 - 1/q)q^k$, obtained as follows: The message is a $k$ dimensional vector over $\Sigma$, denoted $\alpha$. The codeword is indexed by space of $k$-dimensional vectors. The $\beta$-th symbol in the encoding of $\alpha$ is their inner product, that is $\Sigma_{i=1}^k \alpha_i.\beta_i$*

## 3.2 Feige/Langberg/Nissim technique

This paper plays an important part in structure approximation research though the results the authors get out of this paper are of a negative nature. They show that for many well known NP-complete problems, it is NP-hard to produce a solution whose Hamming distance from an optimal solution is considerably closer than getting a solution by taking only a random string [FLN00].

Here they deal with the natural witnesses of the problems. The solution for an optimal value for a NP-complete problem can be called a witness. The witness of a problem is not universal but different problems has different natural witnesses. Suppose, if we consider a SAT problem then the satisfying assignment can be called a witness for that problem. Similarly the bound on the KNAPSACK can be the witness for a KNAPSACK problem and if we consider a Clique problem it can have two witnesses. The set of the vertices by which the clique is created can be a witness and similarly the edges can be marked as 0 and 1 where the edges marked with 1 would form a clique. This can be another witness for the clique problem.

They use many to one reduction in their approach. Such as, from SAT we can get the witness approximation for Clique and so on where Sheldon and Young's [SY03] approach is the turing reduction.

In this paper they takes some of the problems from Karp's list of 21 NP-complete decision problems [citation] and find out their witness approximation(Hamming distance). The following table shows the result of the problems they mention in the paper.

| Problem | Hamming distance |
|---|---|
| 3SAT | $1/2 - \epsilon$ |
| Max-Clique | $1/2 - \epsilon$ |
| Chromatic Number | $2/3 - \epsilon$ |
| Independent Set | $1/2 - \epsilon$ |
| Vertex Cover | $1/2 - \epsilon$ |
| Feedback Edge Set | $1/2 - \epsilon$ (not tight) |
| Directed Hamiltonian Cycle | $1/2 - \epsilon$ |
| Hamiltonian Cycle | $2/5 - \epsilon$(not tight) |

Now we will see the witness inapproximability of SAT which was showed in this paper. They show that given a satisfyable Boolean formula $\Phi$, it is NP-hard to find an assignment $x$ to the variables of $\Phi$ which is of Hamming distance significantly less then $1/2$ to some satisfying assignment of $\Phi$.

**Claim 1.** *Approximating satisfying assignment for 3SAT(and 3CSP) formula within Hamming distance* $1/2 - \epsilon$ *is NP-hard.*

*Proof.* Let $R$= $(w, \phi_0)|\exists a$ s.t $w$=$C(a)$ and $\phi_0(a)$ is true where $\phi_0$ is a SAT formula and the size of the witness $w$ is $n$. They use a technique of finding out the 'core' of the problem and later 'amplify' it. The 'core' of the problem is those variables which are hard to approximate which is $\phi_1$ here. And by 'amplify' they mean the amplification of the 'core' variables to obtain the final formula.

At first using any standard reduction like Cook's theorem, $R$ is transformend into a 3SAT formula. Then the variables of $\phi_1$ is denoted as $w_1, w_2, ....., w_n; z_1, z_2, ..., z_l$, where the $w_1, w_2, ....., w_n$ in a satisfying assignment to $phi_1$ represent a witness $w$ to $phi_0$ in $R$ and $z_1, z_2, ....., z_n$ are the auxiliary variables which are added by the reduction.

Then comes the amplification step where a 3CSP formula $phi_2$ is constructed by duplicating the variables $w_1, w_2, ....., w_n$ so that the number of auxiliary variables $z_1, z_2, ....., z_n$ is small compare to the number of total variable. So, the construction of $phi_2$ looks like $\phi_2(w_1, w_2, ....., w_n; w_1^1, w_2^2, ....., w_n^m; z_1, z_2, ....., z_n)$ and it can be defined as

$\phi_2 = \phi_1(w_1, w_2, ....., w_n; z_1, z_2, ....., z_n) \wedge \bigwedge_{i=1}^{n}(w_i = w_i^1) \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m-1}(w_i^j = w_i^{j+1})$

In some point it contradicts the witness inapproximability of relation $R$ above thus proving the claim.

□

The rest of the problems e.g., Clique, Chromatic number etc are done based on many to one reduction. The remaining problems in the Karp's list which are not discussed here can be done by slight adjustment to the standard reductions. The authors of this paper state that those proofs will apprear in a future version of their paper though it never appeared. So, I made a contact with one of the authors, Dr. Uriel Feige and he told me that partly because of the lack of space and partly because of the technique they used in this paper to prove the witness inapproximability of some of the problem, would produce a proof for remaining of the problems.

# Chapter 4

# Positive results

## 4.1 Symmetric Problems

### 4.1.1 Randomized Algorithm

Randomized algorithms: make random choices during run.

Main benefits:

1.speed: may be faster than any deterministic

2. even if not faster, often simpler (quicksort)

3. sometimes, randomized is best

4. sometime, randomized idea leads to deterministic algorithm

**Example 1.** *Make a string of the same length as the witness string like this it's a binary string so every bit can be either 0 or 1. At first we flip a coin to decide whether the first bit is 0 and then we flip the coin to decide whether the second bit is 0. As we are doing it we will get probability of 1/2 of guessing each bit correct. So our expectation is that we guess about 1/2 of the bits correct and this will give us structure approximation that gets us 1/2 of the bits right.*

### 4.1.2 Weighted Max-Cut example

* So, here is one big question they found that sometimes we can do approximation upto . Weighted max cut is one example where Todd and others were able to give approximation algorithm. Weighted max cut problem is when you have a graph, bunch of vertices , you have weights on edges and you want to cut across the graph so that some of the vertices are on the left some on the right such that the sum of the weight of the edges going across is maximized. Here in the picture if you keep "a" in one side and the rest of the vertices in another then the cut will be maximum. The minimum cut can be solved by network flows but the maximum cut is NP hard. Notice the following if I want to encode this solution into binary string how do I do it? You will have a string of 0s and 1s where for example 0 corresponds to the vertices being on the right and 1 on the left. Notice if you switch 0s with 1s you will get the same solution as both side are the same you just flipped the sides. Now taking arbitrary binary strings we could generate it by flipping a coin and if the coin comes head you put a 1 and if tail then you put a 0. Take a random binary string with a length equal to the number of vertices. This string is guaranteed to have atleast  of the vertices common with the left side or the right side of the optimal solution.

## 4.2 Sheldon and Young's result

# Chapter 5

# Conclusion and future work

# Bibliography

# Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[ACG$^+$99] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization: Problems and Their Approximability Properties.* Springer, 1999.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 2001.

[FLN00] Uriel Feige, Michael Langberg, and Kobbi Nissim. On the hardness of approximating NP witnesses. In *APPROX*, pages 120–131, 2000.

[GHLP99] Anna Gal, Shai Halevi, Richard J. Lipton, and Erez Petrank. Computing the partial solutions. In *14th Annual IEEE Conference on Computational Complexity (CCC'99)*, pages 34 – 45, 1999.

[GR08] Venkatesan Guruswami and Atri Rudra. Soft Decoding, Dual BCH Codes, and Better List-Decodable $\varepsilon$-Biased Codes. In *IEEE Conference on Computational Complexity*, pages 163–174, 2008.

[GR11] Venkatesan Guruswami and Atri Rudra. Soft Decoding, Dual BCH Codes, and Better List-Decodable $\varepsilon$-Biased Codes. *IEEE Transactions on Information Theory*, 57(2):705–717, 2011.

[HMvRW07] Matthew Hamilton, Moritz Müller, Iris van Rooij, and Todd Wareham. Approximating solution structure. In Erik Demaine, Gregory Z. Gutin, Daniel Marx, and Ulrike Stege, editors, *Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs*, number 07281 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany, 2007.

[Hoc97] Dorit S. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* PWS Publishing Co., Boston, MA, USA, 1997.

[KS99] Ravi Kumar and D. Sivakumar. Proofs, Codes, and Polynomial-Time Reducibilities. In *IEEE Conference on Computational Complexity*, pages 46–53, 1999.

[Sud96] Madhu Sudan. Maximum likelihood decoding of Reed-Solomon codes. pages 164–172, 1996.

[Sud00] Madhu Sudan. List Decoding: Algorithms and Applications. *Theoretical Computer Science: Exploring new frontiers of theoretical informatics*, 1872/2000:25–41, 2000.

[SY03] Daniel Sheldon and Neil Young. Hamming approximation of NP witnesses. (Unpublished manuscript referenced in Guruswami/Rudra 2008), 2003.