MATHEMATICS OF COMPUTATION THROUGH THE LENS OF LINEAR EQUATIONS AND LATTICES

MULI (SHMUEL) SAFRA

ABSTRACT

Mathematics of computation and, in particular, *computational complexity theory*, is a fundamental research area in the intersection of computer science and mathematics.

The area revolves around classifying computational problems as feasible or alternatively as infeasible, typically in the worst-case regime.

In some related areas—and even more prominently in practice—the notion of average-case complexity is ubiquitous.

Cryptography is a prime example where proving security of protocols/primitives often necessitates average-case type hardness assumptions.

We take the choice herein to analyze these notions through the lens of *linear algebra*.

This perspective allows us to smoothly present important future research directions, as well as propose conjectures that lay a road-map for future progress.

The goal of this survey is to make research at the core of computation more accessible.

More importantly, it gives us an opportunity to naturally state open questions regarding *lattices*; a solution to which would transform our perception of computation, not only scientifically, but also practically.

MATHEMATICS SUBJECT CLASSIFICATION 2020

Primary 03D15; Secondary 11H06, 11P21, 52C07, 68W25, 11D04

KEYWORDS

Computational-complexity, PCP, lattices, linear-algebra, hardness-of-approximation, cryptography

ICM INTERNATIONAL CONGRESS OF MATHEMATICIANS 2022 JULY 6—14

Proc. Int. Cong. Math. 2022, Vol. 6, pp. 4914–4969
DOI 10.4171/ICM2022/178

1. SYNOPSIS

My presentation will cover central issues regarding the mathematics of computation, in particular the *computational complexity* of algorithmic problems, through the lens of *linear equations*. The aim is to make research at the core of computation more accessible. More importantly, though, I will take the opportunity to present open questions regarding *lattices*. These are research questions whose solutions may transform our perception of computation, not only scientifically, but also practically.

The presentation considers a series of computational problems through the lens of linear algebra, and, more specifically, as problems regarding linear equations. This formalism is flexible enough to provide a uniform perspective across well-known combinatorial optimization problems (Max-Cut is just one example); to clearly present the Unique-Games problem and associated conjecture; and to investigate fundamental advanced problems such as the Shortest-Vector Problem and the Closest-Vector Problem, over codes and, more generally, lattices. Such an approach will help clarify the seemingly subtle key differences between the feasible and infeasible.

We start with the basic framework of a system of linear equations, and the natural computational problem of solving it or at least approximating the solution. The presentation then proceeds through gradual generalizations:

(1) Consider the standard notion where one searches for an assignment to the variables that satisfies as many of the equations as possible, or at least approximating the optimal solution.

(2) Next, refine this notion and consider more general norms by which to measure the quality of an assignment. In the current context, one may think of the linear system as having the form $Mx = t$, and the distance measure being ℓ^0-norm of the difference ($\|Mx - t\|_0 \stackrel{\text{def}}{=} \Pr_i[(Mx)_i \neq t_i]$). Then, extend that notion to a general ℓ^p-norm of the difference between the solution and the target values $\|Mx - t\|_p$.

(3) Consider alternative ways to measure the quality of a solution; a natural one is to assume that the assignment must satisfy all equations, albeit the shorter the assignment, the better.

(4) Finally, further refine the setting by restricting the domain of the assignment. It is natural to define the problem over some field $\mathbb{F}$, and to restrict the domain of the variables to a subset $\mathcal{F} \subseteq \mathbb{F}$.

2. HISTORY

The problem of solving a system of linear equations has been studied extensively since the early history of mathematics—starting with al-Khwarizmi and his invention of linear algebra. The computational, algorithmic question whether a complete solution exists

has long been known to be efficiently solvable via Gaussian elimination. Nevertheless, when changing the definition of the problem slightly, in order to obtain a partial solution of as many of the equations as possible, it turns out that coming up with an optimal solution is hard. In particular, an efficient algorithm that simultaneously solves almost all equations (assuming such a solution exists) would imply that P = NP, resolving the most fundamental open problem in computer science (and one of the most basic open questions of mathematics).

This problem was in fact raised by Hilbert and Ackerman [44] in their Entscheidungsproblem, calling for a systematic algorithm that determines the universality of any given mathematical statement. Much of the basic research into the emerging field of computer science during the 1970s and 1980s was devoted to this conundrum [23, 45, 62]. Consequently, many classical optimization problems were shown to be NP-hard, implying that an efficient algorithm for them would require P = NP.

Let us briefly define these classes of computational (decision) problems: P is the class of all those problems that can be computed efficiently (in polynomial time), while NP is a wider class of such problems and, for our discussion, it suffices to say that it contains many problems that are not known to be in P. An efficient solution for a problem which has been established to be NP-hard would imply that P = NP. The class coNP is the class of (decision) problems whose complement is in NP. The intersection of NP and coNP (problems in NP whose complement is in NP, too) plays an important (if yet not so clear) role in complexity theory, and even more so in cryptography.

An important caveat here is that assuming P ≠ NP is the preferable option for humanity: First, as it relates to the raison d'être of mathematicians, since, if P = NP, there would be no fundamental distinction between the existence of a short mathematical proof and finding it. Second, there are many useful implications to a problem (presumably) being hard, most notably in cryptography, where there is no system or secret safe from the powerful NP class.

The 1980s, through the 1990s and beyond, saw the development of novel techniques designed to facilitate efficient approximation algorithms for such problems. Since coming up with an optimal solution turns out to be unreachable, one naturally attempts to come up with the next best option, that is, a solution that comes close to optimal, namely, within some known *ratio of approximation* from optimal.

Approximation problems could also turn out to be NP-hard. This finding represents the inception of a radically different approach, which requires fundamental novel mathematics, and heralded the dawn of the PCP-era[1] [11, 12, 36]. Indeed, proving the hardness of approximation is vastly more complex and requires utilizing multiple connections to other mathematical fields, including probability, combinatorics, geometry, and analysis. This led to an avalanche of hardness of approximation theorems which established that for many classical optimization problems, coming up with a solution even slightly better than the best-

1 PCP is a strong characterization of the NP class via gap problems (which replace decision problems), thereby allowing proofs that approximation problems are NP-hard.

known is NP-hard. The upshot is that even approximating such a partial solution efficiently would imply the same dreaded consequence that P = NP.

Improving the PCP characterization of NP and proving hardness of approximation problems is a central research topic that has had far reaching implications and is still in flux [5, 7, 14, 17, 19, 26–30, 30, 31, 42, 43, 46, 48, 50, 51, 51–53, 65, 69, 73, 76]. It also has implications that extend beyond complexity, to cryptography, algorithms, and other areas.

Fortunately, most of the fundamental achievements of the PCP-era—where approximation problems are being shown to be hard via the PCP characterization of NP—can be presented clearly through the lens of linear equations. The same is true for the most fundamental open problems of the field of hardness of approximation.

As an immediate consequence, much effort that would have otherwise been spent on trying to come up with approximation algorithms is avoided, as current research indicates that they are unlikely to be efficiently solvable.

The related, but parallel, field of cryptography thrives through an endless quest for reasonable assumptions regarding the hardness of computational problems. This is essential when constructing protocols and primitives with highly intriguing properties that have both theoretical and practical uses, especially with respect to their security. In short, the idea is to hide some secret backdoor information by encoding it into some hard problem (for instance, encoding a bit string as a solution to a system of linear equations), and then to use this encoding to transfer data securely. This begs the question of whether cryptography can be based on the gold standard of P $\neq$ NP.

In this context, it is worth noting that the rich theory of NP-hardness addresses only the worst-case complexity of a problem. Accordingly, an efficient algorithm that is correct on most instances of the problem (that is, on most systems of linear equations) is insufficient. Therefore, an efficient average-case algorithm for an NP-hard problem does not imply P = NP. This is apparently the reason we do not know how to base cryptography on the weakest possible assumption that P $\neq$ NP. A cryptographic protocol chooses keys at random and, if a problem is not known to be hard in the average-case, breaking it would not imply a general algorithm according to the worst-case regime.

Thus, another question naturally arises: Is there a class of computational problems, whose hardness can be established mathematically, similarly to NP-hardness, while maintaining enough flexibility to allow for cryptography? The main topic of interest herein—lattices and lattice-based computational problems—is the prime source of such problems, bringing on a tremendous surge of creative ideas.

2.1. Lattices take center stage

Broadly, a lattice is defined as a discrete subgroup of a topological group, such that the quotient of the group by the lattice satisfies a certain finiteness property. Our interest is in a special case, which we formally define below. The mathematical background on lattices appears in Appendix B.

Historically, lattices have been investigated since the late 18th century by mathematicians such as Lagrange, Gauss, and later, Minkowski (mostly for number-theoretic applications) [68]. Interestingly, some of their theorems (especially Minkowski's) are quite relevant to various parts of our discussion and will thus be presented below in the mathematical foundation section.

In the late 20th century, lattices became a topic of active research with regards to computational aspects of various problems and, more specifically, to the complexity of solving related computational problems. Research in the area has thrived in two directions:

- the quest for algorithms to solve computational problems over lattices as efficiently as possible;
- suggestions of computational problems over lattices as candidates for problems that (presumably) cannot be solved efficiently.

The second point has proved to be more useful than the first, most notably in cryptography, which is always on the lookout for credible assumptions regarding the hardness of computational problems.

In 1982, Lenstra, Lenstra, and Lovasz [LLL or L^3] discovered an algorithm that approximates the Shortest-Vector Problem (henceforth SVP) in a lattice with a weak ratio of approximation (exponential in the dimension). SVP is one of the most fundamental computational problems regarding lattices and it will be covered thoroughly below. The LLL algorithm, which is very useful despite the weak ratio of approximation, has since been utilized for a wide variety of applications. In particular, it may be the first choice as a cryptanalysis tool in an attempt to break a cryptographic system.

In the 1990s, this area was revitalized by a groundbreaking theorem of Miklós Ajtai [4], which established a reduction from worst-case to average-case for a computational lattice problem similar to SVP. He thus turned the spotlight on lattices and, in particular, demonstrated the very interesting advantage that this type of computational problem has when it comes to being presumably hard to compute. To explain this advantage, we need to explain some basic notions in cryptography; we give a brief summary here and more details in the next subsection.

Any modern cryptographic protocol is based on a hardness assumption, that is, a computational problem that is presumed to be hard. The security of the protocol is then established by showing that breaking the protocol entails an efficient solution for the problem in question. But such problems, as we shall explain below, must always be in NP, so if $P = NP$ there would be no cryptography. Hence, any proof of security for a reasonable cryptographic protocol must assume, at the very least, that $P \neq NP$.

For cryptographic applications, however, one needs to rely on a problem that is hard on average, that is, one for which solving a random instance from a prescribed distribution is presumed to be infeasible. The assumption of $P \neq NP$ is not sufficient to guarantee this: most known hardness results for computational problems only guarantee that the worst case of the problem is difficult to solve, and in fact it is unlikely that we can find a problem

that is NP-hard on average, for any probability distribution which can be algorithmically sampled. Remarkably, Ajtai showed that for certain natural lattice problems, the worst case of the problem reduces to the average case. When one has such a worst-to-average-case reduction, one can more comfortably rely on the hardness assumption for the worst-case, and, nevertheless, be able to trust and utilize the hardness of the problem on average.

Consequently, after a remarkable research journey that took more than a decade, Oded Regev [77] eventually introduced the Learning-with-Errors (LWE) problem, which is an average-case problem, and proceeded to establish that it is hard as a computational lattice problem, which can be presumed to be hard in the worst-case. Almost all current cryptography has since been based on the hardness of the LWE problem.

In the next subsection we expand on the preceding paragraphs.

2.2. Hardness assumptions

Let us elaborate on why there can be no modern cryptography without a computational hardness assumption: a protocol suggested for a cryptographic application must have a secret that only authorized entities possess, which facilitates the authorized entities sharing information inaccessible to unauthorized entities (due to their ignorance regarding the secret). But what is the mechanism that keeps the secret secured? Consider the algorithmic problem of finding the secret, assuming access to the data exchanged in the protocol. Modern cryptography, which relies on mathematical proofs of security, assumes full disclosure, that is, that all exchanges are public, and the security of the protocol is established despite adversaries having access to everything but the secret. One must therefore guarantee that it takes way too long to expose the secret, even using strong computers. But a nondeterministic adversary can "guess" the secret and verify the correctness of the guess in similar time (by attempting to decrypt the communication and checking whether the result makes sense). Hence, if $\mathrm{P} = \mathrm{NP}$, for any cryptographic protocol which runs in polynomial time, there exists a polynomial-time algorithm which breaks it.

Since we do not yet know that $\mathrm{P} \neq \mathrm{NP}$, to prove the security of a cryptosystem, we need to *assume* that a certain computational problem is hard, and then show that breaking the cryptosystem is at least as hard as solving the problem. In this case, an algorithm which breaks the cryptographic protocol (that is, exposes the secret, or just learns the secret data) efficiently can also solve the original problem efficiently, which would contradict the hardness assumption. By the above argument, this computational complexity assumption will necessarily be at least as strong as $\mathrm{P} \neq \mathrm{NP}$.

Many problems are by now known to be NP-hard. Thus, assuming they are hard is equivalent to assuming $\mathrm{P} \neq \mathrm{NP}$. Many problems—via PCP—are also known to be NP-hard to approximate. Nevertheless, when focusing on hardness assumptions for cryptographic application, one needs to choose a random secret from a distribution over all possible secrets, and, consequently, the assumption should be related to how hard the problem is to solve on average. This is in sharp contrast to the computational problems whose established NP-hardness results apply only to the worst-case of the problem (that is, how hard it is to solve the hardest instances of a given size). In fact, it is quite unlikely that we can have a problem that is

NP-hard to solve on average (there are reasons to assume that one cannot efficiently identify a subset of instances over which the problem is hard). Consequently, one has to establish security via an assumption stronger than $P \neq NP$ and, furthermore, to come up with an assumption regarding the hardness of the problem over a distribution over the inputs.

Typically, the weakest hardness assumption that enables security is that $P \neq NP \cap coNP$—this may be because one must presume hardness on average, and it is plausible that NP-hard problems are not hard on average for any reasonable, computable distribution. In contrast, some problems in $NP \cap coNP$ are known to be as hard on average as they are hard in the worst case—approximating lattice problems to within a weak enough ratio is one great source for such problems.

Ajtai's brilliant, very insightful theorem was a reduction from approximating a lattice problem (a certain variant of SVP) in the worst case to approximating it in the average case. Consequently, it is enough to assume that this problem is hard in the worst case in order to be able to rely on its hardness on average. The assumption that the problem is hard in the worst case is stronger than $P \neq NP$, but, nevertheless, has proved a rather safe hardness assumption: four decades since its introduction, there is not even a suggestion for an algorithm that could substantially improve on LLL. Moreover, the ratio of approximation that is presumed hard is much stronger (smaller) than that of LLL.

Regev's expansive and wide-ranging perspective on complexity and lattices led him to introduce the Learning-with-Errors (LWE) problem, which assumes a secret vector S that one is trying to discover (i.e., learn). All one can do is press a button, which publicly reveals a random vector x. The rules of the game could be that, in response, the secret holder must announce the inner product of S with x. This, however, would be too easy, as Gaussian elimination can be utilized to reveal the secret S within a small number of such rounds. In LWE, when the button is pushed, x is revealed as before, but what is made public is the inner-product $\langle S, x \rangle$ perturbed by some (small) error. Now, Gauss' algorithm fails, and no other efficient algorithm is known for the problem. Regev has shown a reduction from a lattice problem (a certain variant of SVP) to LWE, thereby establishing a hardness result that is very useful when proving the security of a cryptographic protocol whose hardness relies on the hardness of that lattice problem.

Assuming LWE is hard leads naturally to cryptographic applications. Indeed, the LWE problem yields a natural secret key, S, and a natural public key, the x's and the noisy inner products (we refer to [77] for a complete description of the cryptosystem).

Regev's reduction from a lattice problem to LWE is not only quite complicated but also a quantum reduction, i.e., it implies that LWE is as hard to solve as a shortest-vector-type problem for a quantum computer, but not necessarily for a classical computer. Simplifying the reduction and hopefully avoiding a quantum reduction (that is, finding a classical reduction) is one great research direction to pursue which could greatly improve our understanding of the issues involved and thereby lead to much sought-after progress.

3. FOUNDATION: A SYSTEM OF LINEAR EQUATIONS

Finding a solution for a system of linear equations is one of the most fundamental computational problems. Let us start with definitions:

Definition 3.1 (Linear equations). A system of linear equations consists of a field $\mathbb{F}$, a matrix $M \in \mathbb{F}^{m\times n}$, and a target vector $t \in \mathbb{F}^m$.

The most natural decision problem within this framework is whether a given linear system has a solution that satisfies all the equations. Of course, this is an easy problem, which can be efficiently solved via Gaussian elimination.

Consequently, given a system of linear equations, the next immediate goal is to find an assignment $x \in \mathbb{F}^n$ that satisfies as many of the equations of the system as possible. For instance, given $\Psi = [M, t, \mathbb{F}]$ a system of linear equations, let us denote by $\mathsf{val}[\Psi]$ the minimum, over all assignments $x \in \mathbb{F}^n$, of the fraction of equations that are unsatisfied,

$$\mathsf{val}[\Psi] \stackrel{\text{def}}{=} \frac{1}{m} \min_{x\in\mathbb{F}^n} \left|\left\{i \mid [Mx]_i \neq t_i\right\}\right|.$$

One should remark here that when measuring the computational complexity of a problem, we look at the time it takes to solve it as a function of the length of the input. As the input size grows to infinity, the number of equations could also grow rapidly. It then makes sense to measure the *fraction* of equations unsatisfied. An algorithm that finds an assignment that satisfies 0.99 of the equations (leaves 0.01 unsatisfied), assuming such an assignment exists, solves it for unbounded input size.

The rest of this section surveys numerous variants of algorithmic problems over a system of linear equations, representing the most fundamental open questions in the field of the mathematics of computation. These are variants of the *Closest-Vector Problem (CVP)*—where one is given a target vector t and the algorithmic problems call for a solution coming as close as possible to satisfying the equations. Another important variant is the *Shortest-Vector Problem (SVP)*—where one assumes the target vector $t = \vec{0}$ is all 0, albeit, disallows the all 0 assignment.

The rest of this section surveys numerous variants of algorithmic problems over a system of linear equations. Here is a table of progression, with informal description—the box denotes a computational problem:

- 3.1 **Error correcting codes (ECC)**—the standard setting for computational problems regarding a system of linear equations:
  - **CVP-code search version;** **Compute** 1—decoding a received word by searching for the closest vector.
  - **CVP-code decision;** **Compute** 2—deciding if the received word is acceptable.
  - **SVP-code decision;** **Compute** 3—finding the *distance* of a linear ECC.

  - **Max-Cut;** **Compute** 4—partitioning the vertices of a graph into two sets, maximizing crossing edges.

- 3.2 **Complexity of approximation** considers approximation versions:
  - **CVP gap**$[\varepsilon, 1-\varepsilon]$ **over linear-ECC;** **Compute** 5—distinguishing whether the received word is acceptable or the received word is very far from acceptable.
  - **Unique games (UG)** $\mathtt{PCP}_{1\leftrightarrow 1}[\varepsilon, 1-\varepsilon]$ **Compute** 6—structurally restricted gap-CVP-code. It is an open problem.
  - **2-to-1 games** $\mathtt{PCP}_{2\to 1}[\varepsilon, 1-\varepsilon]$**;** **Compute** 7—almost linear equations (see below for precise definition). It is NP-hard.
  - **Unique games** $\mathtt{PCP}_{1\leftrightarrow 1}[\frac{1}{2}, 1-\varepsilon]$**;** **Compute** 8—a little harder than UG above. It is NP-hard.
- 3.3 **General norm**—one can extend all computational problems mentioned so far to general metrics.
- 3.4 **Alternative measures**
  - **Shortest integer solution;** **Compute** 9—finding the shortest all-satisfying assignment; or at least approximating it.
- 3.5 **Average-case complexity**
  - **Shortest integer solution average case;** **Compute** 10—finding the shortest all-satisfying assignment over a random system of linear equations; or at least approximating it.
  - **Learning-with-Errors (LWE);** **Compute** 11—assuming a secret target vector and, given a random system of linear equations plus a random, noisy version of the secret, finding the satisfying assignment.
- 3.6 **Lattices**—general sparse subdomain (discrete subgroup). For example, including full-rank matrices.
  - **CVP-lattice-search version;** **Compute** 12—finding the closest vector in a lattice.

3.1. Error correcting codes (ECC)

In brief, a *linear error correcting code (linear-ECC)* takes a data vector $x \in \mathbb{F}^n$ and transmits instead the vector $Mx \in \mathbb{F}^m$, where $m \gg n$. The transfer rate of information decreases, as one sends a string of length m instead of length n. Nevertheless, if vectors in the subspace $\{Mx\}$ are distinct enough, assuming not too many errors occur in transmission, the received vector $t \in \mathbb{F}^m$ is not too different from Mx and could therefore be decoded, and

the original vector x extracted. The matrix M is referred to as the *generating matrix* of the linear ECC, and the question becomes whether there is a legal codeword within a certain radius of t, where the radius corresponds to the number of errors that can be handled by the code.

In other words, the decoding problem calls for an almost complete solution for a system of linear equations. Unfortunately, once one allows some equations to not be satisfied, the computational complexity of the problem becomes strikingly different. In terms of linear equations, it goes as follows:

Compute 1 (CVP-code-search). The *Closest-Vector-Search Problem over linear-ECCs* is

- **Input:** a system of linear equations $[M, t, \mathbb{F}]$ and distance $\varepsilon > 0$.
- **Goal:** find a vector generated by M within distance ε from t, $\|Mx - t\|_0 \leq \varepsilon$, assuming one exists. (Alternatively, find the one closest to t).

For a broader perspective, consider the ℓ^0 (Hamming) metric over all vectors $\mathbb{F}^m$ and, within it, the set of vectors that are spanned by M, that is, all vectors of the form Mx for $x \in \mathbb{F}^n$. Since $m \gg n$, the set of such vectors is sparse within the metric and possibly spread thin so that the distance between any pair of vectors is high; hence, finding a vector closest to a general target vector in $\mathbb{F}^m$ is nontrivial.

Our focus throughout is on this type of objects, namely, a discrete sparse subgroup which is sparse within a geometric space over some field. The field could be continuous, for example, $\mathbb{R}$; nevertheless, by default, the set of vectors spanned is discrete, as will become clear below.

The reason this computational problem is classified as concerning ECCs is that the sparse subset is the set of all legal codewords, Mx for $x \in \mathbb{F}^n$. Accordingly, the target can be thought of as the received word and the algorithm's goal is to decode it, namely, find the original transmitted codeword.

When dealing with the computational complexity of a problem, it makes more sense to consider the decision version, which then takes the following form:

Compute 2 (CVP-code). The *Closest-Vector decision problem over linear ECCs* is

- **Input:** a system of linear equations $[M, t, \mathbb{F}]$ and distance $\varepsilon > 0$.
- **Goal:** decide if there is a vector generated by M within distance ε from t—is $\mathsf{val}[M, t, \mathbb{F}] \leq \varepsilon$?

In words, given a system of linear equations, can one satisfy almost all of them?

The CVP code (**Compute** 2) problem turns out to be NP-hard for any not too large $\varepsilon > 0$. Consequently, the above search version (**Compute** 1) of finding the legal codeword of a linear-ECC closest to a given vector t (for a general generating matrix M) is also NP-hard.

A similar, yet quite distinctive, highly important computational problem regarding codes is the Shortest-Vector problem:

Compute 3 (SVP-code). The *Shortest-Vector decision problem over linear ECC* is

- **Input:** a system of linear equations $[M, \mathbb{F}]$ and distance $\varepsilon > 0$.
- **Goal:** decide if there is a *nonzero* vector generated by M within distance ε from $\vec{0}$.

In the linear ECC framework, the smallest ε for which such an x exists is the *distance* of the code.

Max-Cut. Observe that the classical Max-Cut problem is a special case of the CVP-code problem. Recall that an instance of Max-Cut consists of a graph $G = (V, E)$, and the goal is to find a bipartition of the vertices, that is, $V = L \cup R$, such that as many as possible edges go across the cut. Here is a way to phrase the Max-Cut problem as an instance of linear equations: for each vertex $v \in V$, introduce a variable x_v, set the field to be $\mathbb{F}_2$. As for the matrix M, we take it from $\mathbb{F}_2^{m \times n}$ where $m = |E|$, $n = |V|$, and associate each row with an edge and each column with a vertex. We set $M[e, v] = 1$ if the vertex v is an endpoint of the edge e, otherwise set $M[e, v] = 0$. Observe that M's columns are now linearly independent. Finally, let the target vector be $t = \vec{1} \in \mathbb{F}^m$. It is easy to see that for an assignment $x \in \mathbb{F}^n$, the Hamming weight of $Mx - t$ represents the number of edges missing from the cut defined by the bipartition $L = \{v \mid x_v = 0\}$, $R = \{v \mid x_v = 1\}$. Therefore, minimizing this Hamming weight corresponds exactly to maximizing the size of the cut.

Compute 4 (Max-Cut). The *Max-Cut problem* is

- **Input:** a system of linear equations $[M, \vec{1}, \mathbb{F}_2]$ and distance $\varepsilon > 0$; all rows have exactly two 1 entries while all others are 0.
- **Goal:** is there a vector generated by M within distance ε from $t = \vec{1}$? (Alternatively, find the one closest to $\vec{1}$).

Claim A.1 [folklore] below establishes that Max-Cut is NP-hard. Since this is a special case of CVP, the same is true for CVP.

Interestingly, a similar reduction applies when the field is $\mathbb{R}$ and the domain for the solution is $\mathbb{Z}$.

3.2. Complexity of approximation

An approximation algorithm abandons the option of finding an assignment that leaves the smallest possible number of equations unsatisfied. It settles instead for an assignment that leaves a larger fraction of the equations unsatisfied; still, the number of unsatisfied equations is within the ratio of approximation times the optimal.

As it turns out, however, it is hard to distinguish even between the two extreme cases: the case where almost all equations can be simultaneously satisfied versus the case where almost none can be satisfied.

To prove an approximation problem hard, one has to define a problem that is close in structure to a decision problem, nevertheless, whose hardness implies hardness of approx-

imation. Such problems form a *gap problem* where the algorithm is required to distinguish between two extreme cases for the value of the system, but is free to return an arbitrary outcome for the in-between values.

Compute 5 (Gap CVP-code). The *Closest-Vector gap[$\varepsilon, 1-\varepsilon$] over linear-ECCs* problem is

- **Input:** a system of linear equations $[M, t, \mathbb{F}]$.
- **Goal:** distinguish between
  - **Accept:** $\mathsf{val}[M, t, \mathbb{F}] \leq \varepsilon$,
  - **Reject:** $\mathsf{val}[M, t, \mathbb{F}] > 1-\varepsilon$.

Note that in a *gap problem* the algorithm is free to accept or reject all inputs that fall within the gap; in the gap-CVP case, these are equation systems $\Psi = [M, t, \mathbb{F}]$ whose value $\mathsf{val}[\Psi]$ is between the two thresholds, $\varepsilon < \mathsf{val}[\Psi] \leq 1-\varepsilon$.

Claim 3.1. *For any $\varepsilon > 0$, there is a large enough field $\mathbb{F}$ so that CVP-code* $\mathrm{gap}[\varepsilon, 1-\varepsilon]$ *is NP-hard.*

A proof can be found at Claim A.2 below.

We have just established that the gap version of the CVP problem is hard. Numerous other versions of these problems are known to be NP-hard. In fact, for most classical approximation problems, any improved ratio of approximation, beyond the best-known efficiently achievable, is NP-hard—as a consequence of the PCP theorem. There are only a handful of those approximation problems that do not have such a result established, and most of them do have some hardness established—albeit not NP-hardness. Their hardness is relative to a problem whose complexity is not yet established, namely UG—where the hardness of UG (see below) is possibly the most fundamental open problem of the field.

Unique-games (UG). When the equations are restricted to include only two variables—in particular, where every row of the matrix M is zero except for one 1 entry and one -1 entry—it becomes the infamous Unique-Games problem, which is not known to be NP-hard despite strenuous efforts to prove such a statement:

Compute 6 (UG). The *Unique-Games [UG]* $\mathtt{PCP}_{1\leftrightarrow 1}[\varepsilon, 1-\varepsilon]$ problem is

- **Input:** a system of linear equations $[M, t, \mathbb{F}]$ so that every equation is of the form
$$x_i - x_j = t_{ij}.$$
- **Goal:** distinguish between
  - **Accept:** $\mathsf{val}[M, t, \mathbb{F}] \leq \varepsilon$,
  - **Reject:** $\mathsf{val}[M, t, \mathbb{F}] > 1-\varepsilon$.

The syntactic restriction, of every equation consisting of the difference between 2 variables required to be a prescribed value, makes it a special case of CVP and thereby potentially easier than the general form. This opened the door for considerable attacks via sophisticated algorithms, which have succeeded in reducing the time it takes to solve UG to less than a trivial exponential of the size of the input. The infamous *UG conjecture* [47] states that the problem is (NP-)hard to compute.

A grueling effort of more than a decade of work culminated in 2018 in considerable progress on this nearly two-decade-old open problem; the progress concerns the resolution of the related 2-to-1-games conjecture (a problem similar to UG, but where each equation has two optional solutions):

Compute 7 (2-to-1 is hard). The *2-to-1-Games* $\texttt{PCP}_{2\to 1}[\varepsilon, 1-\varepsilon]$ problem is

- **Input:** a set of linear sums $[M, \mathbb{F}]$ where all constraints are of the form
$$x_i - x_j \in \{t_{ij}, t'_{ij}\}.$$
- **Goal:** distinguish between
  - **Accept:** $\mathsf{val}[M, \mathbb{F}] \leq \varepsilon$,
  - **Reject:** $\mathsf{val}[M, \mathbb{F}] > 1-\varepsilon$.

Theorem 3.1 ([53]). *The 2-to-1-Games* $\texttt{PCP}_{2\to 1}[\varepsilon, 1-\varepsilon]$ *is NP-hard.*

As a side note, the (highly complex) proof of Theorem 3.1 starts with hardness of, again, *linear equations* over $\mathbb{F}_2$ [43] and extend it considerably. This has discouraged research attempting to show an efficient algorithm for UG. A fundamental reason for this apparent disinclination is the observation that Theorem 3.1 implies that distinguishing between the value of a UG system of linear equations being $\leq \frac{1}{2}$ versus it being $> 1-\varepsilon$ is NP-hard:

Compute 8 ($\frac{1}{2}$UG). The *Unique-Games* $\texttt{PCP}_{1\leftrightarrow 1}[\frac{1}{2}, 1-\varepsilon]$ problem is

- **Input:** a system of linear equations $[M, t, \mathbb{F}]$ so that every equation is of the form
$$x_i - x_j = t_{ij}.$$
- **Goal:** distinguish between
  - **Accept:** $\mathsf{val}[M, t, \mathbb{F}] \leq \frac{1}{2}$,
  - **Reject:** $\mathsf{val}[M, t, \mathbb{F}] > 1-\varepsilon$.

Corollary 3.2. $\texttt{PCP}_{1\leftrightarrow 1}[\frac{1}{2}, 1-\varepsilon]$ *is NP-hard.*

Proof. First, turn each constraint of $\texttt{PCP}_{2\to 1}$ into two equations, for each of the possible target values. This turns a system of value $1-\varepsilon$ into a $\texttt{PCP}_{1\leftrightarrow 1}$ system of linear equations of value $\frac{1-\varepsilon}{2}$. Then, add a small fraction of obviously satisfied linear equations to take care of the ε. ∎

All the above-mentioned sophisticated algorithms proposed so far for UG, however, produce an assignment satisfying roughly the same fraction of the equations, assuming the value of the system is $\frac{1}{2}$ in the "accept" case or the case where the value is $1-\varepsilon$. Therefore, these algorithms are ill-equipped to resolve the UG-conjecture (unless P = NP) and that direction seems to have been completely abandoned.

3.3. General norm

The next paradigm shift involves altering the measure of the distance of a solution from a perfect solution, by generalizing the ℓ^0 distance (= Hamming distance, that is, the fraction of equations left unsolved) to any norm, for example, the Euclidean ℓ^2-norm or any ℓ^p-norm. This seemingly simple technical change has a considerable impact on the computational complexity of problems, as well as on the mathematics involved.

Cables and wires. Consider the ECC paradigm we have used so far; one can think of it as a way to correct messages transmitted over a cable, which comprises numerous wires. The simplistic approach we have considered so far is that each wire can carry a binary value, namely, can be in either of two options for each clock tic. A more realistic approach is where the wires can carry some specific voltage, depending on the value transmitted, however, there is no reason to limit the number of possible voltages to two. One can have numerous plausible values, each indicating a value of a predetermined range. In that case, the transmitted voltage might come out different at the other end of the wire, nevertheless, it is more likely to change a little than to change by much. Now, recovering the transmitted message, one better utilize this to facilitate a more efficient transmission. The error is thus measured by the distance between the set of voltages transmitted, on each wire, and those received, according to some natural distance (norm). The goal is to figure out the data most likely to have been transmitted, *even if all values have changed (not by much) in transmission.*

In other words, one may take a more general perspective, nonetheless, *over the same object*; instead of looking for the assignment with the smallest number of equations left unsatisfied, which amounts to looking for a vector x so that Mx is within a small radius ℓ^0-*ball* around t. The question becomes how small a radius ball contains a vector Mx, however, *according to a general norm*

$$\mathsf{val}[M,t,\mathbb{F}] \stackrel{\text{def}}{=} \min_{x\in\mathbb{F}^n} \|Mx-t\|,$$

where the norm $\|Mx-t\|$ could be any prescribed norm that the definition of the problem calls for. In other words, take the difference between the prescribed target values t and the actual values the assignment x gives to each sum, and measure them according to some norm such as the Euclidean (ℓ^2) norm or the taxi-cab (ℓ^1) norm, that is, instead of counting how many equations are unsatisfied, measure how close the assignment is to satisfying all the equations.

As a broader perspective, considered as a discrete subgroup of a metric space, the only thing that has changed here is that the metric is now allowed to be any general metric. The set of vectors would still, by default, be a sparse set within the metric.

Consequently, the solution, as well as the computational complexity of the problem, may be different depending on the chosen norm.

Norm over fields. Note that one must assume a well-defined norm over the elements of the field $\mathbb{F}$, and, consequently, over vectors of elements of the field, otherwise the definition would be meaningless. For example, for your typical fields, say $\mathbb{R}$ ($|\alpha|$) or $\mathbb{Z}_q$ ($\min\{|\alpha|, |q-\alpha|\}$ the ℓ^1-norm is naturally defined.

From the ECC perspective, the questions corresponding to classical linear equations ask for a legal codeword closest to a given vector t *according to the Hamming distance*: In the general-norm framework, one changes the distance applied in the last formulation, but otherwise considers the same question: pick your favorite norm ℓ^p, "Is there a legal codeword close enough to the word t?", which results in the following computational question:

$$\exists x : \|Mx - t\|_p \leq \varepsilon?$$

In other words, one may consider the CVP-code-search computational problem from above (**Compute** 1) with the more general definition of val, which, by default, is interpreted over the ℓ^2-norm (even though any other norm could work, and, in fact, ℓ^1 may be more natural in such settings). Accordingly, one could define the more general versions for all the above problems (versions of CVP and SVP).

3.4. Alternative measures

Another alternative type of problems concerning the same object (a system of linear equations) calls for a complete solution for all the equations, albeit optimizing a different parameter. Let us rank distinct all-satisfying assignments according to their norm as vectors $x \in \mathbb{F}^n$, each value being an element of the field. Once the norm of elements of the field has been defined, every assignment has a well-defined norm. We may now consider an algorithmic problem that calls for an all-satisfying assignment while minimizing the norm of the assignment.

The Shortest-Integer-Solution (SIS) problem. The resulting problem—which turns out to be quite important due to the implications discussed below—is the *Shortest-Integer-Solution (SIS) problem* which is another variation on finding a solution for a system of linear equations.

For SIS, the equations must all be solved, and the parameter that measures the quality of the solution is the norm of the assignment, so that the closer those numbers are to 0, the better the solution.

This computational problem is central to worst-to-average-case reductions, and thereby natural, when it comes to hardness assumptions, which facilitates cryptographic security. Indeed, assuming SIS hardness entails the security of some *One-Way-Functions (OWF)* and *Collision-Resistant Hash-functions (CRH)* as presented below.

Let us start with a basic definition. For a matrix over a field $\mathbb{F}$, the shortest nonzero vector in the kernel of M is naturally defined as

$$\mathcal{S}[M, \mathbb{F}] \stackrel{\text{def}}{=} \min_{\substack{\vec{0} \neq x \in \mathbb{F}^n \\ Mx = \vec{0}}} \|x\|.$$

The algorithmic approximation problem searches for a short vector in the kernel of M:

Compute 9 (SIS-worst-case). The *Shortest-Integer-Solution (SIS)* $\mathrm{SIS}[M, \mathbb{F}, \kappa]$ is

- **Input:** a matrix $M \in \mathbb{F}^{m \times n}$.
- **Goal:** find a nonzero assignment $\vec{0} \neq x \in \mathbb{F}^n$ such that $Mx = \vec{0}$ and whose norm is small, namely $0 < \|x\| \leq \kappa$.

For the last part of the definition, relating to the norm of the assignment, one may want to consider the field $\mathbb{Z}_q$, for a prime q, and a norm ℓ_p, in which case the problem is well-defined.

Therefore, the algorithmic problem is to find a relatively short solution, under the assumption that a short solution exists. As will become clear below, coming up with such an algorithm entails a solution to other classical lattice problems and, moreover, to the worst-case versions of those problems, making it natural to assume that SIS is hard.

3.5. Average-case complexity

Average-case complexity is fundamental in terms of complexity theory: we wish we had the proper tools to analyze problems, not in the worst case, but on average, over some prescribed distribution. By default, the complexity of a computational problem (for instance, the time complexity, that is, the number of steps it takes for the algorithm to solve the problem) is regarded as the worst-case complexity: given an algorithm that solves the problem, one measures, for any input length n, how long it takes for the algorithm to solve *any* input of length n. That function, from n to the upper bound on the number of steps, over all inputs of length n, is the time complexity of the algorithm. Average-case complexity, in contrast, calculates the complexity according to the number of steps that would suffice to solve inputs of some length n with high probability, or the average time it would take over inputs (according to some prescribed distribution). Average-case complexity makes more sense in general; unfortunately, our tools are insufficient for a coherent theory of the relationships between classes of average-case problems.

Formally, there is a complexity function $t(n)$ and a small parameter ε so that, for distribution Λ_n over all inputs of length n,

$$\mathop{\mathbb{E}}_{x \sim \Lambda_n} \left[\text{the time it takes for } A(x) \text{ to solve the problem}\right] \leq t(n).$$

Note that the complexity of a problem could be drastically different in the average-case compared to the worst-case. Take, for example, the Factoring problem: given an n-digit number, find out its prime-number factorization. This problem is easy to compute on average over

the uniform distribution, as most numbers have small factors; in contrast, numerous security assertions rely on the problem to be hard in the worst-case, and, in fact, even on average, on some particular distributions. Nevertheless, no distribution is known over which Factoring is hard.[2] Another striking example is SAT. It is one of the basic problems shown hard to compute in the worst-case. On average, some distributions are known to follow a sharp-threshold, namely, where increasing the parameter causes the resulting formula to be satisfiable until some critical parameter when it suddenly becomes nonsatisfiable.

3.5.1. SIS

The first problem considered herein under this regime is SIS; unlike the above definition, let us consider the average-case version of the problem where instances are drawn from a distribution, and the computational complexity relates to how long it takes to solve the problem on average or with high probability:

Compute **10** (SIS-average-case). The *Shortest-Integer-Solution (SIS)* in the average-case regime is a computational problem with parameters $\mathrm{SIS}[n, m, \mathbb{F}, \kappa]$ as follows:

- **Input:** a uniformly random matrix $M \in_R \mathbb{F}^{m \times n}$.
- **Goal:** find a nonzero assignment $\vec{0} \neq x \in \mathbb{F}^n$ such that $Mx = \vec{0}$ and whose norm is small, namely $0 < \|x\| \leq \kappa$.

3.5.2. Learning-with-Errors

This is another average-case problem that is even more important than SIS. In fact, most of the recent—quite fantastic—cryptographic primitives have their proof of security that relies on the (presumed) hardness of this problem.

The Learning-with-Errors (LWE) problem was introduced by Regev in 2005 [77] and is another variant of the system of linear equations problem, in fact, it is in some sense a special case of the CVP-code-search from above.

This problem introduces a twist, as each instance is made up of two parts where one is a randomly chosen secret target vector, while the other is a system of linear equations chosen according to the distribution. The former remains a secret while the latter is made public. The goal of the algorithm is to figure out the secret.

In particular, following the linear equation framework, the secret is a vector $x \in_R \mathbb{F}^n$. Next, a random set of homogeneous linear sums is chosen as well—the matrix M—independently of x, which is made public. Lastly, the evaluation of these sums over the secret vector—only perturbed by some small noise—becomes public, too.

Routinely, one assumes a set of n variables $\vec{x}$, over a field $\mathbb{Z}_q$ for a prime number $q = q(n)$ so that $q \gg n$. In addition, let us assume a (noise) distribution ν over $[-\kappa, \kappa]$ for some $\kappa \ll q$.

---

2 Interestingly, Factoring, in its decision version, is known to be both in NP and in coNP. Hence, it is unlikely to be NP-hard as this would imply NP=coNP.

Compute 11 (LWE, search). The $\mathrm{LWE}_\nu[n, m, \mathbb{Z}_q, \kappa]$ is an average-case problem where

- **Secret:** a chosen secret random vector $x \in_R \mathbb{Z}_q^n$.
- **Input:** a random set of linear sums, namely, a matrix $M \in_R \mathbb{Z}_q^{m \times n}$; further, a vector of *noisy evaluations* $t = Mx + e$ where $e \sim \nu^m$ (that is, one applies the noise ν independently for each linear sum).
- **Goal:** find the secret vector x (given M and t).

The computational goal is to find the secret vector, however, for the approximate version, it suffices to find a vector x' so that $\|Mx' - t\| \leq \kappa$.

An algorithm for CVP-code clearly solves LWE—given M and t as input. This problem is a substantial special case for two reasons: first, it is an average-case problem, and, second, the target vector is not arbitrary, but is rather chosen randomly, via a known noise distribution. Nevertheless, Regev showed LWE to be as hard as the general case (of SVP), thereby providing strong evidence for it being hard, and allowing its (presumed) hardness to be extensively utilized for the purpose of establishing cryptographic security.

One may assume that the number of equations is rather large $m \gg n$ compared to the number of variables; thus, information-wise, there should be a unique solution for x that brings the linear sums close to the noisy evaluations of t. In other words, the algorithm must overcome the error ($e \sim \nu^m$) introduced when choosing t in order to find a vector x' that brings the product with the matrix M not too far from the public (slightly perturbed) target t, i.e., so that $\|Mx' - t\|$ is small.

This problem is shown to be as hard as solving worst-case version of lattice problems, hence, one can rather safely assume that it is hard and, consequently, that it is safe to construct cryptographic protocols whose security relies on the hardness of LWE.

3.6. Discrete subgroup—Lattices

So far, the natural cause for the set of selected vectors being sparse within the general domain is the difference in dimension $m \gg n$. Let us now generalize this notion to produce a natural subdomain regardless of that difference. In particular, one may even consider the cases where the matrix has full rank, namely a set of independent vectors where $m = n$. The computational problems as defined thus far become far easier since there is always a good solution to all equations.

Consequently, let us require the solution (the values assigned to the variables) be not arbitrary values in the field, but rather contained in a restricted sparse subset of the field. The resulting construct is a discrete subgroup, i.e., a lattice that is sparse within the general metric (over $\mathbb{F}^m$); if the subdomain of the assignment is discrete, the resulting subgroup is also discrete, even if the field is not.

In what follows, we assume the matrix is of full rank, for exposition purposes, so that it is clear that the sparseness of a lattice does not stem from the global dimension m being quite larger than the domain of the subgroup. Note, however, that lattices could be similarly defined more generally for non-full-rank matrices.

Definition 3.2. A *lattice* is defined by a (full-rank) matrix (called a basis) $M \in \mathbb{F}^{n\times n}$ over a field $\mathbb{F}$ and a subdomain $\mathcal{F} \subset \mathbb{F}$. The lattice comprises vectors of the form below:

$$\mathcal{L}[M] \stackrel{\text{def}}{=} \{Mx \mid x \in \mathcal{F}^n\}.$$

An extensive mathematical background on lattices appears in Appendix B. We may now consider computational problems over lattices, similar to that we have considered so far. These serve not only as a new perspective on the complexity of computational problems, but also provide further information about the complexity of problems in the average-case. This calls for fundamentally different mathematical analysis and has caused a tectonic shift in cryptography, which is always in need of hardness assumptions for average-case problems on which to base the security of protocols.

The computational problems call for a restricted type of assignments to satisfy the equations. Once the domain of x is sparse within the general domain, the computational and mathematical problems become more interesting. The most natural restriction in general (and in particular throughout our analysis of lattices below) is when the domain of M is the reals $\mathbb{R}$, while the domain of the assignment x comprises the integers $\mathbb{Z}$.

Computational problems over lattices, in particular CVP, consider a lattice plus a target vector $[M, t, \mathbb{F}, \mathcal{F}]$. By default, $\mathcal{F}$ is sparse in $\mathbb{F}$ and $\mathcal{F}$ conforms to $\mathbb{F}$'s structure, for instance, $\mathcal{F}$ could be a ring: the archetypal example is where $\mathbb{F} = \mathbb{R}$ and $\mathcal{F} = \mathbb{Z}$. Now, the distance of the closest vector to a target vector t in this context is the smallest radius of a ball around t which contains a vector of Mx; however, in this case, x is restricted to be a vector over $\mathcal{F}$:

$$\mathsf{val}[M, t, \mathbb{F}, \mathcal{F}] \stackrel{\text{def}}{=} \min_{x\in\mathcal{F}^n} \|Mx - t\|.$$

Restricting the domain and insisting on x taking this form allows the fantastic phenomena (as described extensively below) associated with lattices, in particular the intricate dependency of the computational complexity of problems on the ratio of approximation in quest.

Compute 12 (CVP-lattices). The *Closest-Vector problem*, over $\mathbb{F}, \mathcal{F}$, is as follows:

- **Input:** a lattice basis/matrix $M \in \mathbb{F}^{n\times n}$, a target vector $t \in \mathbb{F}^n$, and distance $\varepsilon > 0$.
- **Goal:** find a vector Mx for some $x \in \mathcal{F}^n$ within distance ε from t, that is, where $\|Mx - t\| \leq \varepsilon$.

This small amendment to the parameter optimized by the algorithmic problem translates to a striking difference with regards to the complexity of these problems. It also diverts the mathematics involved in a completely different direction, opening new and exciting paths of research on interesting relationships between these problems and other variants, as well as on the complexity classes in which they reside.

3.7. Worst-to-average-case

Many classical problems, including some of the above mentioned, are known to be hard, typically NP-hard. Hence, an efficient solution for them implies an efficient solution for

the entire class NP thus P = NP. Some classical problems, however, are not known to be NP-hard, nevertheless, no efficient algorithm is known for them. Their complexity is unclear and are thus a great source of fundamental open research questions regarding their complexity. One option not to be overlooked is that some of these problems may form their own class, which could lie between P (namely, efficiently solvable) and NP-hard. Some of these options will be discussed in the next section.

This, however, is all with respect to a problem's worst-case complexity. When it comes to hardness for the same problem's in the average-case, very little is known. It is unlikely that one could show NP-hardness for an average-case problem, as finding a probability distribution that assigns nonnegligible probability to the set of hard cases is a step towards solving the problem efficiently.

So how does one show hardness on average? For starters, as just observed, one should show hardness relative to a class or a problem different than NP, in particular, relative to a problem unlikely to be NP-hard. Still, what could be a good source for a hard problem that is not NP-hard?

What Ajtai suggested in his groundbreaking paper [4] is to assume that weak approximation of the classical computational problems regarding general lattice (such as CVP and SVP) are hard, and then come up with a *worst-to-average-case reduction*, which would establish that an efficient solution for an average-case problem (such as SIS) entails an efficient solution for a worst-case lattice problem.

Note that this a priori is a surprising concept: reductions from problem A to problem B utilize a procedure for B to solve A. It reads A's input and calls on a procedure for B with instances that encodes A's input so that altogether one can derive a solution for problem A on its input. Here, one is expected to reduce the problem, given an arbitrary input, to random instances that seemingly have no relation to the original input.

Ajtai, in his field-transforming mathematical breakthrough, showed that a version of SVP, the *unique-Shortest-Vector problem (uSVP)* (namely, when $\lambda_2 \gg \lambda_1$) can be reduced to a problem similar to the SIS-average-case problem:

Theorem 3.2 (SIS is hard [4, 66]). *SIS-average-case is as hard as uSVP* [4] *and SIVP* [66].

In terms of mathematical ideas and strategy, this emerging area relies heavily on a program laid out by the great Minkowski; this includes his original theorems, as well as their extensions. These are covered below: the mathematics involved is discussed in Appendix B while some conjectures that could greatly enhance our understanding and abilities in figuring out the complexity of these problems can be found in the next section, as well as in Section 4.3.

Regev's LWE problem. Ajtai's worst-to-average-case reduction to SIS is, nevertheless, not the most natural for applications to cryptography, which prefers computational problems more like CVP—so that the target vector can be kept secret while the matrix is public. One of the most useful theorems—whose applications are highly ubiquitous in cryptography—is the other fundamental worst-to-average-case reduction by Regev [77], who showed a reduc-

tion[3] of a similar lattice problem to the LWE problem. Regev came up with a reduction from GapSVP and SIVP (a lattice problem that computes a set of short linearly independent lattice vectors) to DGS problem—the problem of sampling from the Gaussian distribution over a lattice; he proceeded that with a quantum reduction from DGS to the LWE problem. One of the drawbacks, still a wonder, is that this reduction calls for the Quantum Fourier Transform—which is the basic advantage of quantum algorithms that Shor's algorithm [81] relies on.

By establishing hardness of LWE based on worst-case hardness, Regev constructed a simple and elegant public key cryptography system. Consequently, the security of many (if not almost all) important cryptographic primitives was proved building on this assumption: digital signatures [63], fully homomorphic encryption [20], and many more.

4. PLAN: IN A CLASS OF THEIR OWN

In the grand project of classifying computational problems into feasible or infeasible, one of the remaining fundamental tasks is to identify problems that are in-between, that is, problems that are neither efficiently solvable nor NP-hard. Let us hence define the class of computational problems:

Definition 4.1. The class *NP-intermediate* $\mathtt{NPI} \stackrel{\text{def}}{=} \{A$ unsolvable in polynomial-time and $A \notin$ NP-hard$\}$.

Our understanding of the principles that govern such problems and this class is limited at best. Here is the immediate caveat: if a problem is proven to be in that class, it would imply $\mathrm{P} \neq \mathrm{NP}$; so our goal is to just give some evidence that a problem belongs in that class. The best evidence is if an apparently hard problem is in $\mathrm{NP} \cap \mathrm{coNP}$, thus it being NP-hard implies $\mathrm{NP} = \mathrm{coNP}$. Nevertheless, taking into consideration that some of the problems whose complexity is not yet settled could be in that class, one should develop methodologies to enable such evidences. Some of these problems may be extra related so as to make up a class of their own, in which case one hopes for a computational problem complete for that class. The plan is to identify such subclasses, if they exist, and find more connections between various such problems.

In particular, let us consider approximation problems. Some optimization problems are NP-hard, hence, one tries to approximate them. It is naturally the case that some weak ratios of approximation are known to be feasible while some stronger are known to be NP-hard. For most classical problems, improving even slightly on the ratio known to be feasible is NP-hard; while, for a selected few, there is no known precise ratio of approximation at which the problem immediately switches from feasible to NP-hard.

Computational problems over lattices could serve a crucial role in giving evidence to a problem in or out of the $\mathtt{NPI}$ class and in general exhibiting connections between problems:

3 Albeit, a quantum one—this is a bug not a feature: presuming LWE is found to be easy, this implies an efficient quantum algorithm, not a classical one.

- The first project is to establish up to which ratio these few still unresolved approximation problems are NP-hard, hopefully obtaining tight infeasibility results. This would most likely require novel PCP-reductions, as current technology has not shown any progress on this for nearly two decades.

- Once novel PCP-reductions are developed, these techniques may facilitate reducing apparently hard problems to weaker ratios of approximation for lattice problems, hopefully, to within ratios for which these lattice problems are known to be in coNP (beyond square-root of the dimension). If the reduction manages to cross that threshold, it would prove that the original problem is in coNP, which is the best evidence for not being NP-hard, therefore a great evidence to being in NPI.

- The next project is to extend the scope of the worst-to-average-case reductions. These would require stronger properties of lattices related to Minkowski's theory and, in particular, to reverse Minkowski theorems. With regards to computational problems, it may be quite productive to identify a restricted class of lattices, for which stronger Minkowski-type theorems are true, allowing strong connections with regards to the computational complexity of those problems.

- Lattice-based cryptography relies on the hardness of lattice problems and almost invariably on LWE presumed hard. Richer set of hardness assumptions could broaden and provide more general proofs of security for protocols.

- To complete the picture, note that it is quite possible that some of these lattice problems can be efficiently approximated to within polynomial ratios by a quantum machine (compared to the exponential ratio of the LLL algorithm). Shor's algorithm considers the multiplicative group modulo the number N and utilizes quantum Fourier transform to find cyclic subgroups of it, thereby factoring N. Minkowski developed lattice theory as the "Geometry of Numbers" as it generalizes phenomena related to numbers to high dimensions. It would be only fitting to apply quantum Fourier transform on a lattice $\mathcal{L}$ to find cyclic subgroups within it, thereby finding (or at least approximating) the best basis for the lattice $\mathcal{L}$. If true, this would drastically transform our perspective on compatibility, as the class of feasible computational problems would include much more than those solvable by efficient (random) algorithms. It would mean that the natural process of solving an algorithmic problem would be to translate it into approximating, say, SVP (possibly on average) then applying a quantum machine to solve that.

Let us therefore point out some conjectures and questions that could advance this grand plan. The most fundamental problem in this regards is the Unique-Games [47]. If proven NP-hard, it will immediately imply tight infeasibility results for a large class of approximation problems [51,56,71,73]. Furthermore, it will draw a very strict border of computational infeasibility of approximation and show that, for practically all classical problems,

it is infeasible to improve even slightly on the best known approximation algorithm, assuming $\mathrm{P} \neq \mathrm{NP}$.

Moreover, just improving beyond the current $\frac{1}{2}$ barrier [28, 29, 52, 53] would be a tremendous achievement. Namely, showing that the following algorithmic problem is NP-hard: given a Unique-Games instance promised to be 0.51-satisfiable (namely, of value 0.49), satisfy a nonnegligible fraction of the constraints:

Question 4.1. Is there $c < \frac{1}{2}$, so that $\forall \varepsilon > 0$ there exists an alphabet size $q = q(\varepsilon)$ such that $\mathtt{PCP}_{1\leftrightarrow 1}[c, 1-\varepsilon]$ is NP-hard?

Already here, one of the options that should be given serious consideration is that the Unique-Games problem is not NP-hard, yet neither in P:

Question 4.2. Is there, $\forall \varepsilon > 0$, an alphabet size $q = q(\varepsilon)$ such that $\mathtt{PCP}^{q}_{1\leftrightarrow 1}[\varepsilon, 1-\varepsilon] \in \mathtt{NPI}$?

That is, one may be able to show that the problem is in coNP, which would be quite earth-shuttering:

Question 4.3. Is it true that $\mathtt{PCP}^{q}_{1\leftrightarrow 1}[\varepsilon, 1-\varepsilon] \in \mathrm{coNP}$?

Another manner of giving an evidence of a problem residing in NPI—besides proving a problem is in coNP (as an evidence it is not NP-hard)—is to show connection between problems in that class by reducing one to another.

Thus, naturally, the question arises: For those problems in coNP and NP, how does one establish hardness?

A prime candidate to belong to the class NPI and show it is related to other problems is the canonical Factoring problem, where, given a (binary representation) of a natural number, the algorithmic goal is to find its representation via prime factorization. It is quite likely that this problem is in NPI, furthermore, one should hope that its complexity can be related to lattice problems [1].

Lattices. The complexity of computational problems over lattices is wide open, nevertheless, there are some striking results already established. These problems are interesting in their own right but, furthermore, have the capacity to facilitate results that give evidence to problems being in NPI. Beyond the Factoring problem above, other problems could benefit from being reduced to an approximation variant of a lattice problem.

There could be two purposes for such a reduction: either to establish hardness of the lattice problem, or to show the problem resides in coNP. To show that the lattice problem is hard, one reduces a problem known to be NP-hard to it. To show that the original problem is in coNP, reduce it to a lattice problem, for a weak enough approximation (square-root of the dimension)—these are known to be in $\mathrm{NP} \cap \mathrm{coNP}$. The known complexity results for lattice problems are summarized in Figure 1.

The LLL algorithm [61], including Schnorr's refinement [79] and Babai's extension [15] to CVP, give the exponential upper bound on the right. The hardness for SVP is

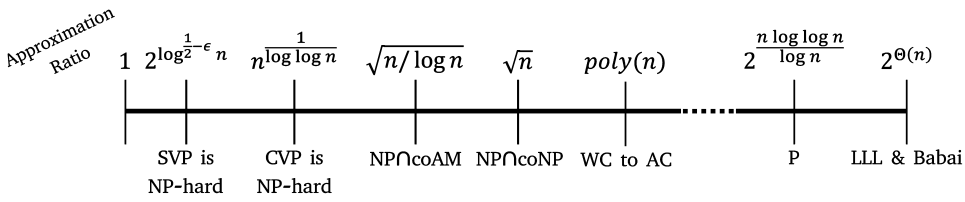

FIGURE 1
Complexity scale of lattice problems for approximation ratios

by Khot [49] following the original result by Micciancio [65]; while the hardness for CVP is from [30].

Let us go on the edge here and conjecture that both SVP and CVP are NP-hard to approximate to within a ratio of almost $\sqrt{n}$:

Conjecture 4.4. *Both CVP and SVP problems are NP-hard to approximate within a ratio of* $\tilde{o}(\sqrt{n})$.

Beyond the $\sqrt{n}$ ratio of approximation, these problems are in coNP[4] (and are clearly in NP) [2, 39]. Thus, Conjecture 4.4 asserts that CVP and SVP are NP-hard almost to the approximation ratio beyond which, if proven NP-hard, it would imply that NP = coNP. Proving these problems are hard almost to the ratios they are known to be in coNP is the holy grail of the field, nevertheless, this direction has seen no progress for almost two decades. A proof for any of these would be monumental and immensely extend our understanding of the complexity of lattice problems. It would also naturally require new types of PCP reductions:

Conjecture 4.5. *Approximating CVP/SVP to within* $c\sqrt{n}$ *(for some global constant* c*) is in* NPI.

These conjectures are all important and with far-reaching consequences. Nevertheless, the connection between classical conjectures (such as UGC) and those related to lattices has not yet been established. Such potential connections are explored next—we discuss some conjectures that could illuminate the tight interplay between classical infeasibility and the complexity of lattice problems.

4.1. Break the soundness barrier

There are two major outstanding open research questions regarding classical gap problems whose resolution would greatly deepen our understanding of the complexity of approximation; these are the unique-games, $\mathrm{PCP}_{1\leftrightarrow 1}$, and 2-to-1-games, $\mathrm{PCP}_{2\to 1}$. There are two major versions for each with distinct characteristics, depending on the placement of the gap: either the assumption is that, in the "accept" case, all constraints are satisfied (also

4 Recall that coNP is the class of problems whose complements are in NP.

known as perfect completeness) or where the assumption is that all but a small fraction are satisfied. In $\texttt{PCP}_{1\leftrightarrow 1}$, when the assumption is that all are satisfied, the problem becomes trivially feasible (via Gaussian elimination). In the $\texttt{PCP}_{2\rightarrow 1}$ problem, when the assumption is that, in the "accept" case, only almost all constraints are satisfied, even satisfying a negligible fraction of the constraints (in the "reject" case) is NP-hard [53].

And thus, the only two versions to further investigate, regarding the "accept" case are:

(1) $\texttt{PCP}_{1\leftrightarrow 1}[\frac{1}{2}-\varepsilon, 1-\varepsilon]$,

(2) $\texttt{PCP}_{2\rightarrow 1}[0,\sigma]$.

Note that the $\texttt{PCP}_{1\leftrightarrow 1}$ problem has been discussed above and, despite the progress made in the last few years, it still seems wide open; as mentioned above, even improving the "accept" case beyond one-half is an open Conjecture 4.1.

Let us then direct our attention to the $\texttt{PCP}_{2\rightarrow 1}$ with perfect completeness. Its complexity is not only fundamentally interesting but—as will be discussed shortly—could prove fruitful in uncovering deep connections between classical PCP and lattice problems, which in turn could have a huge impact on cryptography.

Consider then the stronger promise regarding the "accept" instances, namely, that there is a solution satisfying *all* the constraints. Such a promise makes the problem easier, moreover, adapting the reduction of [53] (for the harder case where the promise is only that there is an assignment satisfying *almost* all constraints) fails fundamentally.

Khot conjectures that $\texttt{PCP}_{2\rightarrow 1}$ is hard even when trying to satisfy a small fraction of the constraints:

Conjecture 4.6 (Khot [47]). *For any field size q, there exists $\varepsilon(q) > 0$, where $\lim_{q\to\infty}\varepsilon(q) = 0$, for which $\texttt{PCP}_{2\rightarrow 1}[0, 1-\varepsilon(q)]$ is NP-hard.*

One should note here that, for σ close enough to 0, there is a folklore theorem indicating that the problem $\texttt{PCP}_{2\rightarrow 1}[0,\sigma]$ is NP-hard (in words, given an instance that can be fully satisfied, finding an assignment that satisfies almost all constraints is NP-hard). Nothing so far is known when the fraction of satisfiable constraints in the "reject" case becomes quite smaller than 1.

And here comes the central question in this context: What is the complexity of the $\texttt{PCP}_{2\rightarrow 1}$ problem when the algorithm needs to distinguish between the case where there is an all-satisfying assignment versus the case where not even a tiny fraction of the constraints can be simultaneously satisfied?

Let us boldly go against the "public opinion" here and suggest:

Conjecture 4.7. *For any alphabet size q, let $\varepsilon_P > 0$ be the largest number so that $\texttt{PCP}_{2\rightarrow 1}[0, 1-\varepsilon] \in P$; there is $\varepsilon_0(q) \gg \varepsilon_P$ so that for all $\varepsilon < \varepsilon_0(q)$,*

$$\texttt{PCP}_{2\rightarrow 1}[0, 1-\varepsilon] \in \text{NP} \cap \text{coNP}.$$

In words, it is clear that all these variants of $\mathtt{PCP}_{2\to 1}$ are in NP; the conjecture claims that once the goal is to satisfy only a very small fraction, the problem could be in coNP as well.

If true, the $\mathtt{PCP}_{2\to 1}$ problem with such parameters could be a perfect computational problem to assume being hard and build on it to prove other problems in $\mathtt{NPI}$ are as hard. As discussed above, cryptography needs problems in NP $\cap$ coNP that can be safely assumed hard, hence, a plausible goal is a worst-to-average-case reduction relying on this hardness assumption.

4.2. Through lattices

Now, how does one go about proving such a statement? The most natural strategy—in light of progress in the last two decades—is to come up with a reduction from this problem to approximating CVP (or SVP) to within a ratio larger (weaker) than the square root of the dimension:

Conjecture 4.8. *Approximating* CVP *to within* $\Omega(\sqrt{n})$ *(where* n *is the lattice dimension) is as hard as solving the* $\mathtt{PCP}_{2\to 1}[0, 1-\varepsilon]$ *for small enough* $\varepsilon < \varepsilon(q)$ *(*q *is the size of the field/alphabet).*

The reason Conjecture 4.8 suffices for proving Conjecture 4.6 is that approximating CVP to within such a ratio is known to be in coNP (and is trivially in NP) [2, 39].

Consequently, one may contemplate a more comfortable hardness assumption to base cryptography on, in particular to base the hardness of LWE on this type of assumption:

Question 4.9. Is LWE as hard as approximating 2-to-1-games? That is, assuming an efficient algorithm for LWE, can one solve $\mathtt{PCP}_{2\to 1}[0, 1-\varepsilon]$ for small enough ε, in particular for any $\varepsilon < \varepsilon_{\mathrm{NP}}$ where $\varepsilon_{\mathrm{NP}}$ is the smallest so that $\mathtt{PCP}_{2\to 1}[0, 1-\varepsilon_{\mathrm{NP}}]$ is NP-hard.

This would immediately allow cryptography to rely on the hardness of 2-to-1-games (albeit, with parameters as stated above) for security proofs. Furthermore, assuming such a reduction is exhibited, one could consider other cryptographic hardness assumptions (preferably those that are not yet established) and try prove them being as hard as 2-to-1-games.

The assumption that the 2-to-1-games problem is infeasible seems more natural, hence the security of a cryptographic protocol based on its worst-case hardness seems more reliable (than being based on the infeasibility of a problem like the unique-SVP). Still, even those who trust the latter assumption would welcome another option for proving security.

Here is an example of such a question: Could the existence of bilinear (or 3-linear) maps be based on the hardness of $\mathtt{PCP}_{2\to 1}$?

4.3. What's hard with lattices?

The fundamental, mathematical principles involved in lattices are crucial in various distinct fields. The theorems proven by Minkowski [67] more than a century ago and the research program he laid out continue to be very relevant. Some aspects affect the complexity

of computational problems over lattices, in particular their average-case complexity, with natural applications to cryptography.

Naturally, the same lattice may be represented by numerous bases. Some properties are abstract and true regardless of the particular basis while some others depend on the basis. From computation's perspective, the lattice is given via a particular basis and the complexity of the computational problem depends fundamentally on the specific basis given as input. If that basis is close to orthogonal, a problem such as SVP could be computed efficiently, while, given an arbitrary basis, the same problem may turn out to be infeasible.

In other words, the hardness of computational problems over lattices stems from the hardness of finding a "good" basis given an arbitrary one. How could one establish such a statement (without proving computational lower bounds)? One could show relative hardness (that is, a reduction) and prove that efficient algorithm for, say, SVP could be utilized to efficiently transform a bad basis for a lattice to the best one, or at least to one that approximates the best one.

One property of a lattice $\mathcal{L}$, which is clearly independent of the choice of basis, is its "orthonormality," that is, how close to orthonormal the "most orthonormal" basis of $\mathcal{L}$ is. For standard lattices—namely, lattices which have a basis that achieve the *successive minima* (see Definition B.8 below)—one could examine this property via the minima: the more similar the lengths of the basis vectors to the minima, the closer to orthonormal the lattice.

Best lattice basis. There are a few distinct (not necessarily equivalent) plausible definitions for the best basis for a lattice; these formulate different manners by which to assert that the basis is either *short* or close as possible to being *orthogonal*. Several options exist for how to define what it means for a basis to be short. It could be the length of the longest vector, the sum of lengths, or the product of lengths. The discussion below is quite abstract, hence could apply to any of these definitions.

Consider the following *greedy* algorithm (not necessarily efficient) to find such a basis: add to the basis the shortest vector in the lattice, and then the next shortest vector, albeit one that is linearly independent of previous basis vectors (that is, not in the span of current basis), etc. Those vectors achieve, in terms of norms, the successive minima. Nevertheless, they do not necessarily form a basis for the lattice; in fact, there are canonical examples of lattices that have no basis achieving the successive minima (any such set of vectors spans a sublattice). Naturally, one should look for a full characterization of nonstandard lattices.

Question 4.10. Is it possible to characterize *nonstandard* lattices, that is, lattices which have no basis achieving their successive minima?

Question 4.11. Let $\mathcal{L}$ be a nonstandard lattice and consider various sequences of linearly independent vectors achieving the successive minima. Is the sublattice $\mathcal{L}' \subset \mathcal{L}$ generated by each of those sequences the same?

Question 4.12. Can one bound from above the length of the vectors in the shortest basis, relative to the successive minima?

Such a characterization could greatly improve our understanding of nonstandard lattices. Moreover, they may facilitate improving our technique regarding compatibility issues.

Another attempt to construct the best basis for a lattice is the (not necessarily efficient) algorithm of Korkin–Zoltriev [60]. The resulting basis is indeed quite short, however, it is clearly not the shortest (whichever reasonable definition of "short" one is interested in): there exists a lattice $\mathcal{L}$ such that we can perform an elementary operation over the Korkin–Zoltriev basis which makes the basis shorter while spanning $\mathcal{L}$.

Therefore, one should first sort out the plausible definition of a global *shortest* basis. Once that has been clarified, the natural question is

Question 4.13. Is CVP/SVP as hard as finding the shortest basis (given an arbitrary one)? Or at least approximating the shortest one?

In reverse. Consider the *discrete-Gaussian* distribution over a lattice with parameter σ as follows. First, choose a random vector according to the Gaussian distribution, with mean 0 and standard deviation σ. Then, round each point in a parallelepiped to the *canonical* vertex of that parallelepiped.

If σ is large enough, the discrete nature of the lattice becomes negligible. That is, the probability of being outside a relatively large ball (large with respect to the standard deviation of the Gaussian) in the discrete-Gaussian is comparable to the Gaussian. A somewhat related phenomenon is that for a large enough σ, if one chooses a random point according to the Gaussian distribution, then, taking it modulo a fundamental parallelepiped, the resulting distribution is very close to uniform.

Worst-to-average-case reductions fundamentally rely on the latter property. This is how the magic of removing any reference to the original input takes place; one can call on the average-case algorithm only giving the vectors modulo the parallelepiped, and use the outcome, knowing the entire vectors.

Stepping back, one should investigate the mathematics of lattices to figure out how large σ needs to be for the Gaussian to satisfy that property (the complexity heavily depends on that stretch). This was addressed by Miccancio and Regev in their fundamental paper [66]. Reverse Minkowski theorems play a central role when trying to either approximate lattice problems, or, alternatively, when trying to translate solving them into another problem, to establish its relevant hardness (these two processes are similar).

Minkowski's convex body theorem [67] states that an upper bound on the determinant of a lattice implies a lower bound on short vectors, that is, that global density implies local density. A reverse Minkowski theorem, proven in [78], states that a lower bound on the determinant of a lattice and its sublattices (intersection of $\mathcal{L}$ with a lattice subspace) implies an upper bound on the amount of short points, i.e., that local density implies global density on a sublattice. Note that local density does not in general imply global density if the lattice is long and narrow. This leads us to define the class of stable lattices.

Definition 4.2. A *stable lattice* $\mathcal{L}$ is a lattice such that $\det[\mathcal{L}] = 1$ and for any $\mathcal{L}' \subset \mathcal{L}$ (intersection of $\mathcal{L}$ with a subspace) it holds that $\det[\mathcal{L}'] \geq 1$.

Theorem 4.1 (Reverse Minkowski [78]). *For any stable lattice* $\mathcal{L} \subset \mathbb{R}^n$, $\rho_{1/t}(\mathcal{L}) \leq 3/2$, *where* $t := 10(\log n + 2)$. *Here, for a lattice* $\mathcal{L} \subset \mathbb{R}^n$ *and* $s > 0$, $\rho_s(\mathcal{L}) := \sum_{y \in \mathcal{L}} e^{-\pi \|y\|^2/s^2}$ *is the Gaussian mass of the lattice with parameter* s.

This theorem allows one to derive a bound on the number of short vectors in the lattice—this is the reason this is referred to theorem as "Reverse Minkowski." Another corollary of that paper is with regards to the covering radius parameter of stable lattices.

Definition 4.3. For a lattice $\mathcal{L} \subset \mathbb{R}^n$, define the *covering radius* $\mu(\mathcal{L})$ as

$$\mu(\mathcal{L}) := \max_{t \in \mathbb{R}^n} \operatorname{dist}(t, \mathcal{L}) = \min\{r \mid \mathcal{L} + \mathcal{B}(r) = \mathbb{R}^n\}.$$

Regev and Stephens-Davidowitz [78] showed that for any stable lattice it holds that $\mu(\mathcal{L}) \leq 4\sqrt{n}(\log n + 10)$. One important open question regarding stable lattices is the Shapira–Weiss conjecture which aims to prove a stronger result.

Conjecture 4.14 ([80]). *Let* $\mathcal{L} \subset \mathbb{R}^n$ *be stable lattice. Then* $\mu(\mathcal{L}) \leq \mu(\mathbb{Z}^n) = \frac{1}{2}\sqrt{n}$.

4.4. Synergy

In summary, the above far-reaching plan includes several huge projects, each of which may prove highly difficult (if true). Putting together all the parts could, nevertheless, prove reformative.

The first part concerns PCP reductions that would establish NP-hardness of approximating CVP/SVP to within the square-root of the dimension—these problems have been open for nearly two decades [30, 49]. Extending that further, the hope is to manage reducing problems in NPI to CVP/SVP with larger (weaker) ratios of approximation, say, some polynomial in the dimension. If true (and proven), such a statement would establish that utilizing a computer that can approximate SVP/CVP to such ratios, one could efficiently solve problems beyond those currently known to have (even randomized) efficient algorithms.

One may interpreted such a result in two ways: first, as implying that trusting the security of a cryptographic protocol relative to such an approximation of SVP/CVP is safe. On the positive side, such reductions could show that a computer that solves the average-case of these problems can solve efficiently problems not necessarily in P (or BPP).

Moreover, one might be able to utilize a worst-to-average-case reduction to be able to trust that average-case SVP/CVP to within some polynomial ratios is hard. The last piece of the puzzle could be a quantum algorithm that actually efficiently solves SVP/CVP, to within such ratio of approximation, on average. This would be a radical transformation of our perspective of computation: If that entire plan is true, it would mean that computation is not an adaptation of mechanical decision making to electronics. It could mean that solving a computational problem would mean translating it to appropriate approximation of lattice problems, on average, then utilizing quantum effects to solve it efficiently.

5. ABOUT COMPUTATIONAL COMPLEXITY AND PCP

The most fundamental open research question of computer science is whether the class P of efficiently solvable problems differs from the class NP, a class that includes many natural problems with no known efficient solution. Almost half a century after research into computational complexity theory has began, we are still quite clueless regarding these general questions. The questions are referred to as *computation lower bounds*, that is, proving that some problem is not computable by some class of computation. Even some weakest models—which are presumed to include only very simple computational problems—are not known to not contain the entire NP class. In light of this fundamental deficiency, a substitute infeasibility proof establishes a problem to be NP-hard, implying that, assuming $P \neq NP$, it is infeasible.

5.1. The dawn of NP-hardness

Cook and Levin [23, 62] and Karp [45] proposed a framework by which to bypass this deficiency in proving computational lower bounds. Their suggestion was to prove that problems are NP-hard, so that any efficient algorithm for the problem would yield an efficient algorithm for the entire class NP. This serves as a conditional proof of infeasibility: if $P \neq NP$, then NP-hard problems are infeasible. Subsequent works have shown many algorithmic problems to be NP-hard. This research field, which has been going strong ever since, is by now the bread and butter of computer science, and is taught in undergraduate courses on the computational complexity theory.

As a result, the focus of research has shifted towards introducing ideas and methodologies by which to tackle NP-hardness. The most natural choice is to design approximation algorithms, that is, algorithms that are guaranteed to find close-to-optimal solutions. This line of research—approximation algorithms—has been carried with various degrees of success. Let us pick two classical approximation problems to demonstrate this with, namely Vertex-Cover and Max-Cut.

For Vertex-Cover, a simple algorithm due to Gavril and Yannakakis[5] guarantees a ratio-2 approximation, while an ingenious algorithm by Geomans and Williamson [38] achieves a surprisingly good approximation for Max-Cut. The algorithm of Geomans and Williamson introduces the roaring power of Semi-Definite-Programs *(SDP)*, a technique that has since become central in the design of approximation algorithms, especially for constraint satisfaction problems.

Despite this considerable progress with respect to the design of algorithms, there was still no way to tell whether a certain approximation ratio for a given problem is attainable efficiently or not, causing researchers to waste precious time in futile attempts to find better approximation algorithms.

5 Papadimitriou and Steiglitz [72] credit this algorithm to Gavril and Yannakakis.

5.2. The PCP era

While it was quite plausible at this point that some approximation solutions are unattainable efficiently, there was no known method to establish this fact, even for those problems for which no known algorithmic techniques could come close to solve.

Fortunately, the Probabilistic Checking of Proofs characterization of NP *(PCP)* was introduced [11, 12, 36]. The intuitive concept—and the reason behind the name—is to model the class NP via a protocol where a weak *Verifier* verifies the validity of a solution. In PCP, the Verifier is allowed to use randomness and sampling, albeit only to read a constant number of bits. It must accept a valid proof with probability 1, and reject an input with no valid solution with probability close to 1. The proof of the PCP characterization of NP is quite complex, involving sum-check, low-degree (Reed–Muller code) test, and the core engine that makes it all happen, namely composition/recursion [12]. Once established, however, one can assume a genesis gap-problem is NP-hard, and proceed to show other problems are NP-hard via gap-preserving reductions.

Subsequent research into PCP and infeasibility produced many other techniques, including the Long-Code [19], Raz's parallel-repetition theorem [76], and the Fourier-analytic framework by Håstad [43] that have all been utilized to prove numerous inapproximability results. In certain cases, the results were optimal, in the sense that they showed that there was no way to improve on the best known algorithm (for instance, [9, 35, 42, 43] among many other works).

An illuminating example. Let us consider a classical example of an NP-hard optimization problem and its approximation version, one that captures the essence of what is known and what is still unknown with respect to the infeasibility of approximation. Define, for a graph $G = \langle V, E \rangle$, the *Max-Independent-Set*,

$$\mathrm{MIS}[G] \stackrel{\mathrm{def}}{=} \max_{S \subseteq V : \forall u,v \in S, \langle u,v \rangle \notin E} \frac{|S|}{|V|}$$

and the natural gap problem

Compute 13 (Gap Max-Independent-Set). The gap-MIS$[\alpha, \beta]$ problem is

- **Input:** a graph $G = \langle V, E \rangle$.
- **Goal:** distinguish between
  - **Accept:** $\mathrm{MIS}[G] \geq \alpha$,
  - **Reject:** $\mathrm{MIS}[G] < \beta$.

Clearly, for a fixed α, as β decreases, the problem becomes computationally easier. The ultimate goal is to classify the algorithmic problems gap-MIS$[\alpha, \beta]$, for all pairs α, β's, into feasible vs. infeasible.

Using the Gavril–Yannakakis algorithm (and observing that a vertex-cover of a graph may be defined as a set of vertices whose complement is an independent-set) one can show that for every $\varepsilon > 0$, the problem gap-MIS$[\frac{1}{2} + \varepsilon, 2\varepsilon]$ can be solved efficiently.

The question of whether the problem gap-MIS$[\frac{1}{2}, \varepsilon]$ is NP-hard epitomizes the limits, goals, dreams, and obstacles of research in the field.

5.3. Fourier transform and analysis of Boolean functions

In his *transformative* paper in the 1990s [43] Håstad was the first to introduce Fourier transform into PCP, in order to achieve optimal analysis of local tests, which in turn allowed him to establish tight infeasibility for numerous fundamental approximation problems (such as MAX-3SAT and Linear-equations over $\mathbb{F}_2$). He also managed to prove the infeasibility, to within $7/6$, of Vertex-Cover. To that end, Håstad proved that gap-MIS$[1/4, 1/8]$ is NP-hard. Consequently, and quite insightfully, he proposed proving infeasibility for a ratio up to 2 for Vertex-Cover, as the most fundamental open question, whose solution will take the field to the next level.

The challenge proposed by Håstad eventually led to quite a complex proof [31, 32], which improves the infeasibility up to a ratio of $10\sqrt{5} - 21 \approx 1.3606$. This result again, follows from an even stronger infeasibility result for the MIS problem, namely, that gap-MIS$[1/3, 1/9]$ is NP-hard. The proof introduces the full scope of the analysis of Boolean functions via Friedgut's Junta theorem [37],[6] as well as several other fields of mathematics [3, 34]. One crucial hidden parameter in that complex proof was consequently brought up to the surface and under the spotlight by Khot.

5.4. The UGC revolution

Since the original PCP theorem was insufficient to establish numerous fundamental problems as infeasible, Khot postulated in 2002 the stronger *Unique-Games conjecture* for that purpose. This breaks the task into two: proving the conjecture, and using it to establish infeasibility for the remaining problems. The latter part has been immensely successful, leading to new algorithmic methodologies, and interesting connection between a priori distinct areas. In addition, many infeasibility results relying on the conjecture have been established. In sharp contrast, there has been little progress with the former part, that of actually proving the conjecture—up until recently, the only debate was whether there had been very little progress, or none whatsoever.

The *Unique-Games* conjecture [47] (abbreviated UGC henceforth) asserts that the UG problem, as defined above **Compute** 6 is (NP-)hard. Khot showed that assuming (a variant of) UGC, for every $\varepsilon > 0$ the problem gap-MIS$[1 - 1/\sqrt{2}, \varepsilon]$ is NP-hard. This result is interesting for at least two reasons:

(1) It is the first infeasibility result for the MIS problem with constant completeness and vanishing soundness (in contrast to Håstad's result and the Dinur–Safra result, where the soundness is constant).

6 This is the first time Friedgut's insightful and widely applied theorem was utilized. Friedgut told us once that till then, he assumed that this theorem will never find application.

(2) It implies that it is NP-hard to approximate the minimum Vertex-Cover of a graph within factor $\sqrt{2} - o(1) \approx 1.41$, improving the Dinur–Safra result.

This result, however, *relies on an unproven conjecture*, stronger than that P $\neq$ NP.

Subsequently, proofs of infeasibility relied on the UGC, and little progress was achieved otherwise with respect to the infeasibility of fundamental approximation problems. The conjecture is very insightful in that it disconnects the basic task of proving the statement, from the corollaries that could be established as a consequence.

The reason UGC is so fundamental has to do with the core methodology that allows a PCP-Verifier to require only a ridiculously small number of reads, namely, the *composition of PCP proofs* [12]. The natural route by which to prove the UGC is to start with a $\mathtt{PCP}_{1\leftrightarrow 1}$ with not so good parameters—say, $\mathtt{PCP}_{1\leftrightarrow 1}[\varepsilon_a, \varepsilon_r]$ for $0 < \varepsilon_a \ll \varepsilon_r \ll 1$—then use the composition/recursion to improve those parameters while maintaining the restriction on the constraints. The problem is that no such NP-hardness has been established to start the process. Another route would be to try to establish the special structure of the constraints.

Postulating the conjecture proposed a two-part plan: *UGC-proof*—prove the conjecture; and *UGC-use*—show infeasibility assuming that the UGC is true. Until recently, little to no progress had been made on the UGC-proof part. On the UGC-use part of the plan, in contrast, research has flourished, and numerous tight infeasibility results were established based on the UGC (only a small selection of which can be mentioned here). Assuming the UGC, Khot and Regev [55] proved that it is NP-hard to approximate Vertex-Cover to within a ratio $2 - o(1)$, that is, the trivial algorithm mentioned above is optimal.

As to the other problem mentioned above, in 2005, Khot, Kindler, Mossel, and O'Donnell [51] proved that the ingenious algorithm of [38] for Max-Cut is optimal in that any improvement over it would refute the UGC. That paper [51] relied on another conjecture called "Majority is most stable," which, gaining extra motivation, was subsequently proven in [71]. These insights made the community realize that there may be a close connection between UGC and SDP algorithms.

Many consequences of UGC followed (cf. [13, 22, 41, 54, 57] to name a few). Remarkably, assuming UGC, it was shown in [73] that a generic Semi-Definite Program gives the best approximation algorithm for the general class of Constraint-Satisfaction problems *(CSP)*. In other words, Raghavendra proved that the connection between UGC and SDP algorithms is very tight and, assuming UGC, the best approximation algorithm for any CSP is SDP based. Studying UGC also extended to other fields, and led to interesting developments in analysis, geometry [70, 71], integrality-gaps, and metric-embedding [24, 56].

UGC has also provoked a large body of work on algorithms [8, 10, 17, 21, 40, 47, 59, 83] designed to refute the conjecture. Despite the failure of these attempts, the ideas therein motivated the study of new algorithmic techniques (such as the Lasserre Semi-Definite Programs hierarchy) and uncovered surprising connections, between problems and among concepts (e.g., between small-set expansion and UGC [8, 74]).

The conjecture has further led to a large body of work introducing new connections and many new ideas; for attempts at proving it, refuting it (many exciting new algorithmic

ideas were introduced for that purpose), and showing NP-hardness assuming it to be true. This area of the field has seen tremendous activity in the last decade and a half.

Summary. While there has been impressive progress on the UGC-use part of the plan, with respect to the UGC-proof part, the UGC remains open despite immense efforts, leaving computer science's best researchers perplexed. We have recently managed considerable progress on the UGC-proof part, as discussed next.

5.5. Present and future

The 2-to-1-Games conjecture, proposed by Khot (in the same paper as the UGC) assumes a less restrictive structure regarding satisfying assignments of PCP: once the PCP-Verifier has read the first value, it can accept 2 values for the second item read (in contrast to the single value accepted in the Unique-Games). This makes the problem harder to solve, and therefore the conjecture that it is infeasible is weaker. Nevertheless, the two statements are quite related—best demonstrated by the fact that there is a trivial reduction from 2-to-1-Games (**Compute** 7) to 1-to-1-Games (**Compute** 8) with value $\frac{1}{2}$ in the accept case.

Only quite recently, Khot, Minzer, and Safra have achieved significant progress, establishing that the 2-to-1-Games problem, albeit with nonperfect completeness, is in fact NP-hard. This achievement has been recognized by the community as going halfway towards the Unique-Games conjecture. The proof is a culmination of a series of four publications [28, 29, 52, 53].

Some consequences of the newly established hardness of 2-to-1-Games are the *unconditional* NP-hardness of gap-MIS$[1 - 1/\sqrt{2}, \varepsilon]$ (establishing an unconditional infeasibility result for the gap-MIS with constant completeness and vanishing soundness), and NP-hardness of approximating minimum Vertex-Cover of a graph within factor $\sqrt{2} - o(1)$ (improving on the Dinur–Safra result).

The first paper [52] presented a new approach for proving infeasibility for the *nonstandard* 2-to-1-Games as well as for Vertex-Cover. The proof relies on a careful combination of sophisticated PCP techniques (such as smooth-parallel-repetition, composition, and the biased-long-code) and a novel component in this context, namely, the Grassmann graph—the only construct we know that allows weak 2-to-1 tests to be shown NP-hard.

The second paper [29] extended the ideas and proposed a candidate reduction, which, unlike the first, could work for *standard* 2-to-1-Games. This approach relies on a combinatorial statement regarding the Grassmann graph of a type not explored before. The third paper [28] initiated that investigation. It was clearly necessary to first study *expansion* on the Grassmann graph (just in time, it was shown by [18] that it is also sufficient). We utilized spectral analysis, which is considerably more challenging to carry out here than on the binary hypercube. However, we still did not manage to prove the statement necessary so as to complete the project: the more ambitious the statements we were trying to prove, the messier the analysis got (to the point where any improvement would take weeks or even months of case analysis). The final piece in the puzzle came in [53], where we completed the plan as

set out by [28] and proved the necessary combinatorial statement, thus completing the grand program as set by [52].

Border of feasibility. This theorem immediately implies tight infeasibility for several fundamental problems, in the sense that improving even slightly on the best known algorithm is impossible (unless P = NP). Furthermore, it implies that given a Unique-Games instance that is promised to be $\frac{1}{2}$-satisfiable, it is infeasible to efficiently come up with an assignment that satisfies even a negligible fraction of the constraints.[7] This in turn has dramatic implications, since all known algorithms, when applied to $\frac{1}{2}$-satisfiable instances, are guaranteed to perform as well as on the standard $(1-\varepsilon)$-satisfiable instances. Therefore, all known algorithmic techniques can be disqualified: if they were to solve the standard Unique-Games problem efficiently, they would also solve the nonstandard, promised to be $\frac{1}{2}$-satisfiable instance and thereby the 2-to-1-Games problem. Because the known algorithms have proved to be inadequate, researchers who still doubt UGC are now challenged to introduce new algorithmic methodologies. This is a win–win situation since it will either produce new algorithmic methods, or else the UGC will be positively resolved.

One of the central tools employed in these endeavors is the analysis of Boolean function, via Fourier transform and related techniques. This facilitates proofs for optimal local-to-global properties with regard to local-tests of binary error-correcting codes, such as Hadamard-code or the Long-code. While of independent interest, much of the extra motivation for that research relates to applications for infeasibility.

5.6. Related open problems

Small-set expansion. Assume a graph $G = (V, E)$ and a set of vertices $S \subseteq V$: the *edge expansion* of S, denoted by $\Phi(S)$, is the ratio of the number of edges inside S to the total number of edges adjacent to any vertex in S.

The *Small-Set Expansion* problem, denoted by $\mathsf{SSE}_\delta(\varepsilon, 1-\varepsilon)$, asks to distinguish between the case that a given graph has a set containing δ fraction of the vertices whose *edge expansion* is at most ε, and the case that any such set has edge expansion at least $1-\varepsilon$. The problem was first suggested as a possible step towards tackling UGC: Raghavendra and Steurer proved that if the Small-Set Expansion problem is NP-hard, then the UGC is true [74]. Subsequent work has shown that the hardness of SSE implies hardness of various problems not captured by UGC (cf. [64, 75])

Interestingly, as yet, the recent progress on UGC has not had any affect on our understanding of SSE.

Question 5.1. Is there $\beta < 1$ such that for any $\varepsilon > 0$ there is $\delta > 0$ such that $\mathsf{SSE}_\delta(\beta, 1-\varepsilon)$ is NP-hard?

---

7 This is why the community refers to it as going $\frac{1}{2}$ the distance towards the Unique-Games conjecture.

Extreme tests and how to prove the UGC. All PCP constructions have a tensor structure, and eventually rely on error-correcting codes with special properties such as local-testing and local-correcting. An important aspect by which the recent proof of the 2-to-1-Games conjecture deviates from previous PCP techniques is with respect to list-decoding properties. Prior to our recent work, the codes that were utilized were list-decodable, that is, any word that passes the local-test with noticeable probability *corresponds* to a short list of legal-codewords. By "corresponds" we mean that except for negligible probability, local values that pass the test must agree with one of the codewords from the short list. Our work, in contrast, utilizes the Grassmann code (along with the Grassmann test), which is not list-decodable in the traditional sense. Nevertheless, a much weaker list-decodability property does hold for the Grassmann code, one which, nonetheless, is sufficient to prove hardness for 2-to-1-Games. In the Grassmann test, any subgraph that is by itself a Grassmann graph could potentially be assigned arbitrary values. Despite this complication, sophisticated analysis does establish some weak notion of global consistency, fortunately, one that suffices for the proof of the conjecture.

Going beyond "list-decodable codes" appears to be a prerequisite to prove infeasibility for d-to-d-Games (for constant d). It is possible that an even weaker notion of decodability might be necessary in order to prove the Unique-Games conjecture. For the Unique-Games conjecture, properties similar to those of the Grassmann code must hold for the desired code and, furthermore, its local-tester should be a 1-to-1 test. It is quite plausible that the decodability notion of this code would have to be even weaker, while still allowing some form of global consistency to be established. It is quite possible that the key insight here is to extend that notion so as to make the claimed notion of global consistency even weaker. We believe that a more sophisticated statement, similar to that regarding the Grassmann graph, has the potential to break the barrier and finally succeed in going below the 2-to-1 barrier.

The conjecture has been tantalizing the best researchers for almost two decades, with little progress made until recently. Nevertheless, in light of recent progress [53] where we proved the closely related, yet weaker, *2-to-1-Games* conjecture—recognized by the community as going half the distance toward the Unique-Games[8] conjecture— this project seems to have a better potential to succeed than ever before.

6. DISCUSSION

The purpose of this survey is to examine basic questions regarding the mathematics of computing through the lens of computational problems over linear equations in general and lattices in particular. The goal is to shed light on some fundamental open research questions regarding the mathematics of computing and facilitate progress on solving them. The plan and related conjectures might turn out to be false, nevertheless, a solution whichever

8 The Unique-Games can also be referred to as 1-to-1-Games. The 2-to-1-Games problem is another variant of the general d-to-d-Games, as detailed below.

way could prove transformative. On the other hand, if all parts turn out to be true, this could change our vision with regards to computation, not only theoretically but possibly also with practical applications.

First, one may attempt at showing a lattice computational problem to be infeasible (NP-hard) to the extreme ratio, which is assumed to be $O(\sqrt{n})$.

This is likely to require PCP theorems beyond current methodology. The consequences of such progress may facilitate a solution to the 2-to-1-Games problem, $\mathrm{PCP}_{2\to 1}[0, 1-\varepsilon]$,[9] proving it is reducible to weaker ratios of approximation for lattice problems and therefore placing it in coNP for nontrivial ratios (ratios not known to be NP-hard or computationally feasible). This would be a strong evidence for this problem not being NP-hard. Furthermore, it may lead to worst-to-average-case reductions that would in turn yield improved cryptography through richer infeasibility assumptions.

To that end, one should clearly pursue research on the intriguing and beautiful mathematics of lattices and error-correcting codes, including new, improved algorithms or, alternatively, establishing currently known algorithms (such as LLL) as optimal. One should also pursue improved theorems of the reverse Minkowski type, to establish the weight of points on the lattice according to Gaussian measures (of various standard deviation). Such theorems may require further research of the Fourier transform over lattices. Lastly, one should pursue tighter connections between hardness of computational problems over lattices and codes, in the worst as well as average case.

The last piece of the plan would be to try and come up with efficient quantum algorithms for approximating those lattice problems with the mentioned ratios, thereby broadening the scope of feasible computational problems to some subclass of NP ∩ coNP [33].

ACKNOWLEDGMENTS

This survey could not have been written without the involvement and highly helpful remarks of the following, to whom I am very grateful: Dor Minzer, Subhash Khot, Zeev Rudnik, Wojtek Samotij, Itamar Rot, Yael Eisenberg, Oren Renard, Ori Petel, and Eli Putterman.

The author was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 835152), and by an ISF grant and a BSF grant.

9 Let us mention here that, in contrast, $\mathrm{PCP}_{2\to 1}[\varepsilon, 1-\varepsilon]$ has been proven to be NP-hard [53], which, in words, says that given a 2-to-1-Games instance, promised to have an almost perfect assignment, satisfying even a small fraction of the constraints is NP-hard. The proof uses practically all the basic mathematical machinery invented for such purposes and some more. The main mathematical breakthrough was to prove an expansion structure theorem on functions similar to Boolean functions. The reduction there relies fundamentally on the imperfect completeness, as it starts with linear equations over $\mathbb{F}_2$ which are known to be NP-hard only for imperfect completeness (and are efficiently solvable with perfect completeness).

A. THE COMPLEXITY OF CVP

Computational problems regarding linear equations may be solvable, NP-hard, or in-between. The outcome is interesting regardless of the option, and could allow great insight into complexity of classes, in particular average-case complexity, with deep and wide implications.

A.1. CVP is NP-hard

To see that CVP is NP-hard, consider the Max-Cut problem reformulated as equation instead of graph.

Claim A.1 (Decision Max-Cut is NP-hard). *The decision problem Max-*Cut$[1-\varepsilon]$ *(which accepts if the best assignment satisfies* $\geq 1-\varepsilon$*) is NP-hard for small enough* ε.

Proof. Reduce from 3SAT, assuming p clauses over n variables, each occurring 5 times in distinct clauses (which one can easily translate into an instance satisfying this assumption).

Variables.

- 5 variables for each clause: $y_{i,j}$ for $i \in [p]$ and $j \in [5]$—one for each occurrence of a literal in each clause, corresponding to $j = 1, 2, 3$, plus two auxiliary variables for every clause, where $j = 4, 5$, altogether $5p$.
- Extra (ghost) variables, so that every original variable x_i has exactly 5 positive and 5 negative variables associated with it. Name them $x_{i,j,b}$ for $i \in [n]$ and $j \in [5]$ and $b \in \{0, 1\}$ with the caveat that some of the x's already appear as y's.
- 1 global variable y_F.

Equations.

- Between all opposing occurrences of same variable, namely, for any $x_{i,j,0}$ and $x_{i,j',1}$ for $j, j' \in [5]$ (that is, 5 by 5 combinations), have an equation
$$x_{i,j,0} + x_{i,j',1} = 1.$$
- For any clause C_i, we have an equation for every pair $(y_{i,j}, y_{i,j'})$ where $j \neq j'$,
$$y_{i,j} + y_{i,j'} = 1.$$
- Between y_F and all variables except the ghosts ($5m$ of the clauses variable—5 $y_{i,j}$ for each clause)—satisfied only if those variable have a different value than y_F,
$$y_{i,j} + y_F = 1,$$
so the global variable y_F forms a clique of size 6 with any 5 variables $y_{i,j}$ for a clause $j \in [p]$.

The total number of equations is $25n + 15p$.

Completeness. A satisfying assignment to the original 3CNF can be translated to an assignment satisfying $25n + 9p$ of new equations (out of $25n + 15p$):

- Assign $y_F = 0$.
- Satisfy all 5 by 5 for each original variable, assigning those 0/1 in accordance to the value the satisfying assignment to the 3CNF gives it.
- Assign the auxiliary variables for each clause 0/1 values that ensure that exactly 3 of the 5 are assigned 1 (for each clause, at least one of the literals is 1, so the auxiliary two can always guarantee 3 that are 1).

Altogether $9p$ out of the $15p$ are satisfied, $3 + 3 \cdot 2$.

Soundness. First, observe that the maximum is achieved when all $25n$ variable equations are satisfied (the number of equations satisfied, if not split according to sign, is ≤ 20 of the 25). Then, notice that out of the 15 for every clause, at most 9 can be satisfied: 3 by 3 split of the clique of size 6 formed with y_F.

So to complete the proof, observe that in each clause, to ensure that 9 of the equations are satisfied, at least one of the original literals must be 1, to satisfy 3 of the equations against y_F. ■

Now, as to the approximation:

Claim A.2 (Gap Max-Cut is NP-hard). *The gap problem Max-*Cut$[1-\varepsilon(1+?), 1-\varepsilon]$ *(which accepts if the maximal assignment satisfies* $\geq 1-\varepsilon$ *and rejects if it is* $< 1-(1+?)\varepsilon$*) is NP-hard.*

Proof. Reduce from gap-E3SAT$[7/8 + \varepsilon, 1]$, which is NP-hard. ■

B. FOUNDATIONS OF LATTICES

Let us formally define a lattice and, for clarity, discuss for now only full-rank lattices, where the dimension and the column-rank are equal. Accordingly, we will assume from now on that the columns of a basis/matrix $M \in \mathbb{F}^{n \times n}$ are linearly independent.

The fact that the columns of the matrix are linearly independent implies that they form a basis for the vector space $\mathbb{F}^n$. Denote the ith column of a matrix M by M_i, so one can write $M = \{M_1, \ldots, M_n\}$ to specify the vectors of the basis, and Mx denotes the linear combination of the basis' column vectors.

In the early (code) versions of the computational problems above, the sets of vectors $\{Mx \mid x \in \mathbb{F}^n\}$ formed a sparse subspace of $\mathbb{F}^m$. This sparseness is due to the number of variables being smaller than the dimension m of the entire space ($n \ll m$). In our case, since the matrix is square and $n = m$, how can we expect the set to be sparse and not to span all $\mathbb{F}^n$? It is indeed a crucial aspect of lattices that the domain $\mathcal{F}$ of the variables $\vec{x}_i$'s is sparse within $\mathbb{F}$. The set of vectors of a lattice form a sparse vector space within the entire space $\mathbb{F}^n$. The target vector could be any vector in $\mathbb{F}^n$, and one looks for a vector close to it

of the form Mx, where $x \in \mathcal{F}^n$ is a vector in the lattice. Throughout, we denote the domain of all $\vec{x}_i$ by $\mathcal{F} \subset \mathbb{F}$, and assume it to be a ring with respect to the same operations as $\mathbb{F}$.

Definition B.1 (Lattice). Given a field $\mathbb{F}$ and its restriction $\mathcal{F} \subset \mathbb{F}$, a *lattice* is defined by a matrix $M \in \mathbb{F}^{n \times n}$ as

$$\mathcal{L}_{\mathbb{F}|\mathcal{F}}[M] \stackrel{\text{def}}{=} \left\{ Mx \mid x \in \mathcal{F}^n \right\}.$$

Note that we assume $\mathcal{F}$ forms a discrete ring within $\mathbb{F}$, assuming a natural metric over $\mathbb{F}$.

Our primary focus herein is the case where the domain of the matrix is $\mathbb{F} = \mathbb{R}$, and thus $M \in \mathbb{R}^{n \times n}$, while the domain of the ring comprises the integers, $\mathbb{Z}$, and thus $x \in \mathbb{Z}^n$,

$$\mathcal{L}[M] \stackrel{\text{def}}{=} \left\{ Mx \mid x \in \mathbb{Z}^n \right\},$$

nevertheless, some of the statements below are true for more general settings.

A more intuitive perspective of such a formalism is a sparse vector space within $\mathbb{R}^n$, determined by a basis (the columns of the matrix M) while allowing only integral linear combinations of the basis' vectors

$$\mathcal{L}[M] \stackrel{\text{def}}{=} \left\{ \sum a_i M_i \mid a_i \in \mathbb{Z} \right\}.$$

Observe that, because the domain $\mathcal{F}$ of the variables $\vec{x}$ being sparse, different bases for the same vector space $\mathbb{F}^n$ do not necessarily span the same lattice.

Here is an examples of a 2-dimensional lattice:

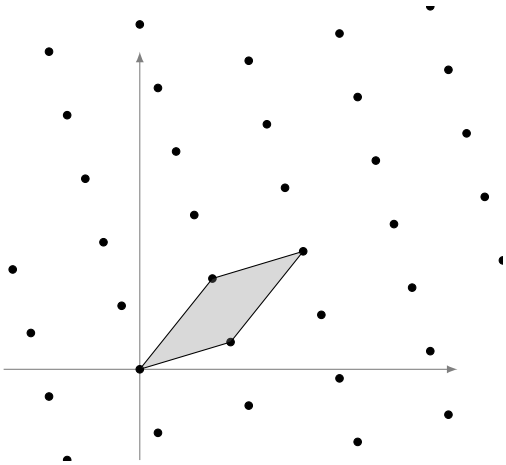

FIGURE 2
The lattice spanned by $(1, 0.3)$ and $(0.8, 1)$

B.1. Elementary, my dear

The same lattice can be generated by an infinite number of bases. A natural question is how one can determinate whether two bases generate the same lattice.

Claim B.1. *Let $M_1, M_2 \in \mathbb{R}^{n\times n}$ be full-rank matrices. Then $\mathcal{L}[M_1] = \mathcal{L}[M_2]$ iff there exists a unimodular matrix $U \in \mathrm{GL}_n(\mathbb{Z})$ such that $M_2 = \mathrm{U}M_1$.*

Invariance. In fact, any lattice basis can be transformed to a different basis of the same lattice by a sequence of elementary column operations: swapping columns, multiplying a column by -1, or adding an integer multiple of a column to another column.

Duality is quite an important concept in general, and here in particular: every lattice has a unique *dual lattice*.

Definition B.2 (Dual lattice). For a lattice $\mathcal{L}$, the lattice

$$\mathcal{L}^\vee \stackrel{\text{def}}{=} \left\{u \in \mathbb{R}^n \mid \forall v \in \mathcal{L} : \langle u, v\rangle \in \mathbb{Z}\right\}$$

is referred to as the *dual lattice* of $\mathcal{L}$.

Definition B.3 (Dual matrix). Given a full-rank matrix $M \in \mathbb{R}^{n\times n}$, its *dual matrix* is

$$M^\vee \stackrel{\text{def}}{=} M^{\top -1}.$$

Fact B.2. *The dual of the basis is a basis for the dual lattice, $\mathcal{L}^\vee[M] = \mathcal{L}[M^\vee]$.*

Trivially note that any lattice is a dual of its dual:

Fact B.3. $\mathcal{L}^{\vee\vee} = \mathcal{L}$; $M^{\vee\vee} = M$.

The following definition deals with the *stretch* of a lattice by some factor

Definition B.4. Given a lattice $\mathcal{L}$, for $q \in \mathbb{R}^+$, let $\mathcal{L}[M]$'s *q-scaled lattice* be

$$q \cdot \mathcal{L} \stackrel{\text{def}}{=} \{q \cdot v \mid v \in \mathcal{L}\}.$$

Fact B.4. $q \cdot \mathcal{L}[M] = \mathcal{L}[q \cdot M]$.

Another interesting fact regrading the relation between the stretched dual of the lattice and the dual of the dual is the following

Fact B.5. $(q \cdot \mathcal{L})^\vee = q^{-1} \cdot \mathcal{L}^\vee$.

The definitions so far relate to an abstract lattice, with no particular basis. Let us now examine some notions that are specific to a particular basis for a lattice.

Fundamentals. The *fundamental parallelepiped* of a lattice basis is the body defined by letting all x_i coefficients be between 0 and 1:

Definition B.5 (Fundamental parallelepiped). For a matrix M, define the *fundamental parallelepiped* as

$$\mathcal{P}[M] \stackrel{\text{def}}{=} \{Mx \mid \forall i,\ 0 \le x_i < 1\}.$$

Note that the fundamental parallelepiped tiles space when we shift it by a lattice vector, as it is half open.

Definition B.6. Given a matrix M and a vector $v \in \mathbb{R}^n$, the *rounding* of v according to the fundamental parallelepiped, denoted as $\lfloor v \rfloor_{\mathcal{P}[M]}$, is defined by

$$\lfloor v \rfloor_{\mathcal{P}[M]} \stackrel{\text{def}}{=} \sum_i \lfloor a_i \rfloor M_i \ ,$$

where the coefficients a_i are the unique representation of v according to the basis given by the columns of M, namely $v = \sum_i a_i M_i$.

The mapping $v \to \lfloor v \rfloor_{\mathcal{P}[M]}$ can be thought of as a rounding function, mapping v to some lattice point in $\mathcal{L}[M]$ that is "closest to it" in terms of the coefficients.[10] Looking at the difference then, one gets a point from the fundamental parallelepiped $\mathcal{P}[M]$ (which could be thought of as the "fractional part" of v). We refer to this operation as modulo $\mathcal{P}[M]$, formally defined as

$$v \ \left(\text{mod } \mathcal{P}[M]\right) \stackrel{\text{def}}{=} v - \lfloor v \rfloor_{\mathcal{P}[M]} \in \mathcal{P}[M].$$

Clearly, different bases (for the same lattice) result in different fundamental parallelepipeds and consequently have different rounding values. This could affect the distance of the modulus vector.

Let us now use this notion of the fundamental parallelepiped to establish a method by which to figure out whether two matrices span the same lattice:

Lemma B.6. *Let $M, M' \in \mathbb{R}^{n \times n}$ be full-rank matrices so that $\mathcal{L}[M'] \subseteq \mathcal{L}[M]$ (equivalently, every column M'_i is a vector in $\mathcal{L}[M]$); then*

$$\mathcal{L}\big[M'\big] = \mathcal{L}[M] \iff \mathcal{P}\big[M'\big] \cap \mathcal{L}[M] = \{\vec{0}\}.$$

Proof. Observe that any vector $v \in \mathbb{R}^n$ can be written as $v = \sum_i a_i M'_i$ for $a_i \in \mathbb{R}$.

($\Longrightarrow$). By definition, $\mathcal{P}[M']$ includes only vectors v where all coefficients are $0 \leq a_i < 1$ while any nonzero vector in $\mathcal{L}[M'] = \mathcal{L}[M]$ has at least one coefficients $|a_i| \geq 1$.

($\Longleftarrow$). To show that $\mathcal{L}[M] \subseteq \mathcal{L}[M']$, consider a vector $v \in \mathcal{L}[M]$; by definition, the vector $\lfloor v \rfloor_{\mathcal{P}[M']} \in \mathcal{L}[M'] \subseteq \mathcal{L}[M]$, and *since a lattice is closed under vector addition and negation*, the difference between those two, $v \pmod{\mathcal{P}[M']}$, is in $\mathcal{L}[M]$. On the other hand, $v \pmod{\mathcal{P}[M']} \in \mathcal{P}[M']$ and, by the premise, it must then be the case that $v \pmod{\mathcal{P}[M']} = \vec{0}$ and thus $v = \lfloor v \rfloor_{\mathcal{P}[M']} \in \mathcal{L}[M']$. ■

Determinant. What is the volume of the fundamental parallelepiped?

Recall that the volume of the parallelepiped $\mathsf{vol}[\mathcal{P}[M]] = |\mathsf{det}[M]|$.

Fact B.7. *Given two matrices $M, M' \in \mathbb{R}^{n \times n}$, if $\mathcal{L}[M] = \mathcal{L}[M']$ then $|\mathsf{det}[M']| = |\mathsf{det}[M]|$.*

---

10 Observe that v could be very far from $\lfloor v \rfloor_{\mathcal{P}[M]}$.

This implies that the volume of the fundamental parallelepiped of a given lattice is independent of the choice of basis. This parameter is referred to as the determinant of a lattice.

Definition B.7 (Determinant). The *determinant* of a lattice $\mathcal{L}[M]$ is defined as

$$\det\big[\mathcal{L}[M]\big] \stackrel{\text{def}}{=} \mathrm{vol}\big[\mathcal{P}[M]\big] = \big|\det[M]\big|.$$

The determinant of a lattice is inversely proportional to its density: the larger the determinant, the sparser the lattice. Consider a ball of some (large) radius r around the origin and count how many points of the lattice are in that ball; it follows that the smaller the determinant, the more lattice points are in the ball,

$$\lim_{r\to\infty} \frac{|\mathcal{L} \cap \mathcal{B}[\vec{0}, r]|}{\mathrm{vol}[\mathcal{B}[r]]} = \frac{1}{\det[\mathcal{L}]}.$$

One of the most fundamental questions regarding lattices is when r becomes large enough for the value to become negligibly different from the limit. For a basis/matrix M that is close—in some reasonable sense—to an orthonormal basis, and the value converges quickly; in general, there is a nontrivial upper bound (established below) which is central to this theory.

The denser the $\mathcal{L}$, the less dense the $\mathcal{L}^\vee$, which can be summarized by the following:

Claim B.8. *For a lattice* $\mathcal{L}$, $\det[\mathcal{L}]\det[\mathcal{L}^\vee] = 1$.

Successive minima. Consider the ℓ^2-ball of radius r around the origin

$$\mathcal{B}[\vec{0}, r] \stackrel{\text{def}}{=} \big\{v \mid \|v\|_2 \le r\big\}$$

and the smallest r such that $\mathcal{B}[\vec{0}, r]$ contains a set of vectors of $\mathcal{L}$ which spans an i-dimensional space:

Definition B.8. $\lambda_i[\mathcal{L}] \stackrel{\text{def}}{=} \inf\{r \mid \dim[\mathrm{span}[\mathcal{L} \cap \mathcal{B}[\vec{0}, r]]] \ge i\}$.

Orthonormalization. Orthonormlization[11] is the process of taking a sequence of linearly-independent vectors and replacing it with an orthonormal basis, so that any prefix of the sequence spans the same space as the original prefix. The orthonormal basis is unique.

Definition B.9. Let $u_1, \ldots, u_n \in \mathbb{R}^n$ be a sequence of linearly-independent vectors. The *orthonormalization of* $u_1, \ldots, u_n$ is the sequence $u_1^\perp, \ldots, u_n^\perp$ so that

- $u_1^\perp, \ldots, u_n^\perp$ is an orthonormal basis.
- For $1 \le i \le n : u_i^\perp \in \mathrm{span}[u_1, \ldots, u_i] \setminus \mathrm{span}[u_1, \ldots, u_{i-1}]$.

Fact B.9. *$u_1^\perp, \ldots, u_n^\perp$ is unique (up to signs).*

Observe that the resulting basis, $u_1^\perp, \ldots, u_n^\perp$, is not necessarily a basis for the same lattice.

Denote by $M^\perp$ the matrix whose columns are $u_1^\perp, \ldots, u_n^\perp$.

---

11 This is the orthonormal version of the Gram–Schmidt orthogonalization: all vectors are of norm 1.

Regarding λ_1. One can immediately bound from below the first $\lambda_1[\mathcal{L}[M]]$ successive minimum—the length of the shortest vector—according to $M^\perp$:

Theorem B.10. *Let M be a basis for a lattice $\mathcal{L}$, and $M^\perp$ be its orthonormalization. Then*

$$\lambda_1\big[\mathcal{L}[M]\big] \geq \min_i \big|\langle M_i, M_i^\perp\rangle\big| > 0.$$

Proof. To bound the norm of any nonzero lattice vector from below, let us show a stronger claim: for any coefficient vector $z \in \mathbb{Z}^n$, it is the case that $\|Mz\| \geq |\langle M_k, M_k^\perp\rangle|$, where k is the maximal index so that $z_k \neq 0$. Note that the inner product $\langle Mz, M_k^\perp\rangle$ is independent of all indices other than k (larger indices are 0 and $M_k^\perp$ is orthogonal to $\operatorname{span}[M_1, \ldots, M_{k-1}]$):

$$\begin{aligned}\|Mz\| = \|Mz\|\big\|M_k^\perp\big\| \geq \big|\langle Mz, M_k^\perp\rangle\big| &= \left|\left\langle \sum_{i=1}^{k} z_i M_i, M_k^\perp \right\rangle\right| = \big|\langle z_k M_k, M_k^\perp\rangle\big| \\ &= |z_k|\big|\langle M_k, M_k^\perp\rangle\big|,\end{aligned}$$

where the inequality is due to the Cauchy–Schwarz inequality; now, observe that z_k is a nonzero integer to complete the proof.

Hence, we have just shown that the inner product of the kth vector with its orthonormal basis vector is a lower bound on the shortest nonzero vector in $\mathcal{L}$. ∎

Stretch and inclusion.

Claim B.11. *For an n-dimensional lattice $\mathcal{L}$, there exists a sequence of linearly-independent vectors $u_1, \ldots, u_n \in \mathcal{L}$ so that $\forall i, \|u_i\| = \lambda_i[\mathcal{L}]$.*

Furthermore, for any such sequence it holds that

$$\mathcal{B}\big(\vec{0}, \lambda_{i+1}[\mathcal{L}]\big) \cap \mathcal{L} \subseteq \operatorname{span}[u_1, \ldots, u_i],$$

that is, the intersection of the open ball of radius λ_{i+1} with $\mathcal{L}$ is spanned by the first i vectors.

Banaszczyk. Let us mention here the transference theorem by Banaszczyk, which is a far less obvious statement regarding the successive minima of a lattice and its dual:

Theorem B.1 (Banaszczyk). $1 \leq \lambda_1[\mathcal{L}]\lambda_n[\mathcal{L}^\vee] \leq n$.

Proof. For the first inequality, let s be the shortest in $\mathcal{L}$, and $u_1, \ldots, u_n \in \mathcal{L}^\vee$ be a linearly-independent set of vectors so that $\|u_i\| = \lambda_i[\mathcal{L}^\vee]$ (which exists due to Claim B.11). Then u_i is a basis of $\mathbb{R}^n$. Therefore, there must be an i for which $\langle s, u_i\rangle \neq 0$, and due to duality, $1 \leq |\langle s, u_i\rangle| \leq \|s\|\|u_i\| = \lambda_1[\mathcal{L}]\lambda_i[\mathcal{L}^\vee] \leq \lambda_1[\mathcal{L}]\lambda_n[\mathcal{L}^\vee]$. For the proof of the second inequality, see [84]. ∎

B.2. Minkowski

Theorem B.12 (Blichfeldt). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and a measurable set $S \subset \mathbb{R}^n$ of $\mathrm{vol}[S] > \det[\mathcal{L}]$, there must be $u \neq v \in S$ with $u - v \in \mathcal{L}$.*

Proof. Let M be a basis for $\mathcal{L}$. Note that we can break S according to the shift of the fundamental parallelepiped $S = \dot{\bigcup}_{w\in\mathcal{L}} S_w$ where

$$S_w \stackrel{\text{def}}{=} \{v \in S \mid \lfloor v \rfloor_{\mathcal{P}[M]} = w\} \quad \text{and} \quad \mathsf{vol}[S] = \sum_{w\in\mathcal{L}} \mathsf{vol}[S_w].$$

As shifting by a vector is measure-preserving and $\mathsf{vol}[S] > \mathsf{vol}[\mathcal{P}[M]]$, there must be two distinct points in S that are the same modulo the parallelepiped, that is, a pair $u \neq v \in S$ so that $u \in S_w, v \in S_{w'}$ for $w \neq w' \in \mathcal{L}[M]$ and

$$u \pmod{\mathcal{P}[M]} = v \pmod{\mathcal{P}[M]},$$

and thus

$$u - v = \lfloor u \rfloor_{\mathcal{P}[M]} - \lfloor v \rfloor_{\mathcal{P}[M]} = w - w' \in \mathcal{L}[M]. \qquad \blacksquare$$

Theorem B.13 (Minkowski's convex-body theorem). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and a centrally symmetric convex body $S \subset \mathbb{R}^n$ with $\mathsf{vol}[S] > 2^n \det[\mathcal{L}]$ there must be $u \neq \vec{0}$ so that $u \in \mathcal{L} \cap S$, in other words, $\mathcal{L} \cap S \neq \{\vec{0}\}$.*

Proof. Let $S_{\frac{1}{2}} \stackrel{\text{def}}{=} \{v \mid 2v \in S\}$ and observe that

$$\mathsf{vol}[S_{\frac{1}{2}}] = 2^{-n}\mathsf{vol}[S] > \det[\mathcal{L}],$$

and thus, by Theorem B.12, there must be $u \neq v \in S_{\frac{1}{2}}$ with $u - v \in \mathcal{L} - \{\vec{0}\}$. By definition, $2u, 2v \in S$ and since S is centrally-symmetric, so is $-2v \in S$ and by convexity

$$u - v = \frac{1}{2}(2u - 2v) \in S,$$

therefore, $u - v \in \mathcal{L} \cap S \setminus \{0\}$. $\blacksquare$

Theorem B.14 (Minkowski's second theorem). *For any $\mathcal{L} \subset \mathbb{R}^n$,*

$$\prod_{i=1}^{n} \lambda_i[\mathcal{L}] \leq 2^n \frac{\det[\mathcal{L}]}{\mathsf{vol}[\mathcal{B}[\vec{0}, 1]]}.$$

Proof. Consider a set of linearly-independent vectors $u_1, \ldots, u_n \in \mathcal{L}$ so that $\|u_i\| = \lambda_i[\mathcal{L}]$ (which exists due to claim B.11 above) and let $u_1^\perp, \ldots, u_n^\perp$ be their orthonormalization. Let us define the convex symmetric body (in fact, an ellipsoid)

$$\mathcal{E} \stackrel{\text{def}}{=} \left\{ v \in \mathbb{R}^n \mid \sum_{i=1}^{n} \frac{\langle v, u_i^\perp \rangle^2}{\lambda_i[\mathcal{L}]^2} < 1 \right\}.$$

Now, show

Claim B.15. $\mathcal{E} \cap \mathcal{L} = \{\vec{0}\}$.

Proof. The differences between the open balls of radius λ_{k+1} and λ_k for $k = 0, \ldots, n$ (where $\lambda_0 \stackrel{\text{def}}{=} 0$ and $\lambda_{n+1} \stackrel{\text{def}}{=} \infty$) partition $\mathbb{R}^n$, thus, for $u \in \mathcal{L} \setminus \{\vec{0}\}$ there must be a unique $k > 0$ so that

$$u \in \mathcal{L} \cap \big(\mathcal{B}(\vec{0}, \lambda_{k+1}) \setminus \mathcal{B}(\vec{0}, \lambda_k)\big).$$

By Claim B.11, $u \in \text{span}[u_1, \ldots, u_k]$ and thus

$$\lambda_k[\mathcal{L}]^2 \le \|u\|^2 = \sum_{i=1}^{k} \langle u, u_i^{\perp} \rangle^2 \le \lambda_k[\mathcal{L}]^2 \sum_{i=1}^{k} \frac{\langle u, u_i^{\perp} \rangle^2}{\lambda_i[\mathcal{L}]^2} < \lambda_k[\mathcal{L}]^2,$$

and it must be the case that the only lattice point in $\mathcal{E}$ is $\vec{0}$. ■

This, in turn, implies, by Minkowski's Theorem B.13 above, an upper bound on T's volume

$$2^n \det[\mathcal{L}] \ge \text{vol}[\mathcal{E}] = \text{vol}\big[\mathcal{B}(\vec{0}, 1)\big] \prod_{i=1}^{n} \lambda_i[\mathcal{L}],$$

which completes the proof. ■

The volume of a ball can be found here.

Corollary B.16 (Minkowski's first theorem). *For any $\mathcal{L} \subset \mathbb{R}^n$, $\lambda_1[\mathcal{L}] \le (\det[\mathcal{L}])^{\frac{1}{n}} \sqrt{n}$.*

Dual lattice. The latter inequality implies some meaningful statements regarding the relation between a lattice and its dual:

Claim B.17. *For any (full-rank) lattice $\mathcal{L}$, $\lambda_1[\mathcal{L}] \cdot \lambda_1[\mathcal{L}^{\vee}] \le n$.*

Proof. By Minkowski's bound, Corollary B.16,

$$\lambda_1[\mathcal{L}] \cdot \lambda_1\big[\mathcal{L}^{\vee}\big] \le \sqrt{n} \cdot \det[\mathcal{L}]^{\frac{1}{n}} \cdot \sqrt{n} \cdot \det\big[\mathcal{L}^{\vee}\big]^{\frac{1}{n}} = n \cdot \big(\det[\mathcal{L}] \cdot \det\big[\mathcal{L}^{\vee}\big]\big)^{\frac{1}{n}}.$$ ■

B.3. Fourier transform over lattices, etc.

Fourier transform. It is defined (see [82]) as follows:

Definition B.10. The Fourier transform of a function $f \in L^1(\mathbb{R}^n)$ is a function $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ defined as

$$\hat{f}(w) \stackrel{\text{def}}{=} \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, w \rangle} \, dx.$$

Claim B.18. *Some basic properties of Fourier transform:*

(1) $\widehat{f + g}(w) = \hat{f}(w) + \hat{g}(w), \widehat{\lambda \cdot f} = \lambda \cdot \hat{f}(w).$

(2) *If $h(x) = f(x + z)$ then $\hat{h}(w) = e^{2\pi i \langle w, z \rangle} \hat{f}(w)$.*

(3) *If $h(x) = f(\lambda x)$ then $\hat{h}(w) = \frac{1}{\lambda^n} \hat{f}(\frac{w}{\lambda})$.*

(4) *If $f(x) = f_1(x_1) \cdots f_n(x_n)$ then $\hat{f}(y) = \widehat{f_1}(y_1) \cdots \widehat{f_n}(y_n)$.*

Definition B.11 (Gaussian). Denote by $\mathcal{N}_n[\sigma]$ the *Gaussian* (normal) n-dimensional (namely, over $\mathbb{R}^n$) *distribution* centered at $\vec{0}$ and of "width" (standard-deviation) σ.

$$\mathcal{N}_n[\sigma](x) = \frac{1}{(2\pi)^{\frac{n}{2}} \sigma^n} e^{-\frac{1}{2\sigma^2} \|x\|^2}.$$

Fact B.19. *The Fourier transform of the Gaussian distribution $\mathcal{N}_n[\sigma]$ is the Gaussian distribution $(\frac{\sqrt{2\pi}}{\sigma})^n \mathcal{N}_n[1/\sigma]$.*

The Poisson summation formula. It asserts the following:

Fact B.20. *For any "nice" $f : \mathbb{R}^n \to \mathbb{R}$ and for any full-rank n-dimensional lattice $\mathcal{L}$,*

$$\sum_{v \in \mathcal{L}} f(v) = \frac{1}{\det(\mathcal{L})} \sum_{v \in \mathcal{L}^\vee} \hat{f}(v).$$

Altogether this gives

Theorem B.21. *For any full-rank n-dimensional lattice $\mathcal{L}$ and any $\varepsilon > 0$, there is $r(\varepsilon)$ so that*

$$\Pr_{v \sim \mathcal{N}_n[\lambda_n[\mathcal{L}] \cdot r(\varepsilon)]} \left[v \notin \mathcal{B}(\lambda_n) \mid v \in \mathcal{L}^\vee \right] \leq \varepsilon.$$

B.4. The Korkin–Zolotarev basis

Let us consider the projection of a lattice $\mathcal{L}$ according to a given vector $u \in \mathcal{L}$.

Definition B.12. For a vector $u \in \mathcal{L}$ of a lattice $\mathcal{L}$, denote by

$$\pi_{\perp u}(v) \stackrel{\text{def}}{=} v - \frac{\langle u, v \rangle}{\|u\|^2} u,$$

which altogether gives the $(n-1)$-dimensional lattice, the result of projecting $\mathcal{L}$ on the hyperplane perpendicular to u, thus

$$\mathcal{L}_{\perp u} \stackrel{\text{def}}{=} \{\pi_{\perp u}(v) \mid v \in \mathcal{L}\}.$$

Definition B.13. A matrix M is a *Korkin–Zolotarev basis* [60] for $\mathcal{L}[M]$ if

- M_1 is shortest, $\|M_1\| = \lambda_1[\mathcal{L}[M]]$;
- $\pi_{\perp M_1}(M_2), \ldots, \pi_{\perp M_1}(M_n)$ is a Korkin–Zolotarev basis for $\mathcal{L}_{\perp M_1}$;
- For any $1 \leq j < i \leq n$, $\langle M_i^\perp, M_j \rangle \leq \frac{1}{2} \langle M_j^\perp, M_j \rangle$.

Fact B.22. *The Korkin–Zolotarev basis always exists (constructively).*

B.5. The LLL algorithm

Recall the orthonormalization process described above (the orthonormal version of Gram–Schmidt) of Definition B.9.

Obviously, applying the orthonormalization process to a given lattice basis does not necessarily result in another basis for the lattice. This is the most fundamental problem with applying the orthonormalization process as is to a lattice basis.

Let us therefore try a similar process that preserves the lattice spanned by the basis, in exchange for a weaker notion of orthogonality. In other words, the inner product between any two vectors must not be too large and the angle between the vector and the span of previous vectors is within $[\pi/3, 2\pi/3]$, instead of being $\pi/2$.

There is also the issue of the order in which the process is carried out, that is, the order of the columns of M:[12] the algorithm and the outcome of the orthonormalization

12 Recall that order does not change at all the lattice spanned by the basis.

process depend strongly on the order of the columns in M. Ideally, one would be fantastically able to find an order according to which the projection $\langle M_i, M_i^{\perp}\rangle$ is monotonically nondecreasing, in which case it follows from Theorem B.10 that M_1 is the shortest vector. Unfortunately, the requirements of quasiorthogonality and nondecreasing projection are generally mutually exclusive.

The condition below is a relaxation of the nondecreasing order and represents a compromise, an order in which the next vector projection is at least a constant times the current. Unfortunately, this constant turns the algorithm into an approximation algorithm and, moreover, one with only an exponential ratio of approximation!

Let us therefore define an LLL-reduced basis [61]:

Definition B.14. A basis/matrix M is *δ-LLL-reduced* (think of $\delta \in (\frac{1}{4}, 1)$) if the following two properties hold:

- *(Close to orthogonal)*

$$\forall 1 \le j < i \le n, \quad \left|\langle M_j, M_j^{\perp}\rangle\right| \ge 2\left|\langle M_i, M_j^{\perp}\rangle\right|,$$

  that is, there is no way to add or subtract M_j from M_i to make them more perpendicular.

- *(Lovász condition)*

$$\forall 1 \le i < n, \quad \delta\left\langle M_i, M_i^{\perp}\right\rangle^2 \le \left\langle M_{i+1}, M_i^{\perp}\right\rangle^2 + \left\langle M_{i+1}, M_{i+1}^{\perp}\right\rangle^2.$$

To motivate the extra term in Lovász condition, take, for example, two vectors of length 1, so that the angle between them is $\pi/3$. The orthogonality condition holds, but $|\langle M_2, M_2^{\perp}\rangle| = \sqrt{\frac{3}{4}}$. Due to symmetry, however, changing the order of columns changes nothing. Therefore, the extra term guarantees that, if M_i is switched with M_{i+1} and M_i is adjusted to be as orthogonal as possible, there would be no reason to switch back (preferring the vector with smaller norm).

Fact B.23. *For a δ-LLL-reduced basis M, for any $1 \le i < n$,*

$$\left\langle M_{i+1}, M_{i+1}^{\perp}\right\rangle \ge \sqrt{\delta - \frac{1}{4}}\left\langle M_i, M_i^{\perp}\right\rangle.$$

Claim B.24. *Let M be a δ-LLL-reduced basis, then*

$$\lambda_1\big[\mathcal{L}[M]\big] \ge \|M_1\|\left(\frac{\sqrt{4\delta - 1}}{2}\right)^{n-1}.$$

Proof. First, note that, for any $1 \le i \le n$, applying Fact B.23 $(i-1)$-times (and observing that $\|M_1\| = \langle M_1, M_1^{\perp}\rangle$) implies

$$\left\langle M_i, M_i^{\perp}\right\rangle \ge \left(\delta - \frac{1}{4}\right)^{\frac{i-1}{2}} \|M_1\|,$$

which achieves its minimum when $i = n$, thus, by Theorem B.10 above,

$$\lambda_1\big[\mathcal{L}[M]\big] \ge \min_i \left|\langle M_i, M_i^{\perp}\rangle\right| \ge \left(\delta - \frac{1}{4}\right)^{\frac{n-1}{2}} \|M_1\|. \qquad \blacksquare$$

B.5.1. The algorithm

Algorithm 1: The LLL algorithm

Input: full-rank matrix M and $\delta \in (\frac{1}{4}, 1)$
Output: δ-LLL-reduced basis for $\mathcal{L}[M]$
repeat
  Compute orthonormalized $M^{\perp}$
  for $i \leftarrow 2$ **to** n; $j \leftarrow i-1$ **to** 1 **do**
    $M_i \leftarrow M_i - \lfloor \frac{\langle M_i, M_j^{\perp} \rangle}{\langle M_j, M_j^{\perp} \rangle} \rceil M_j$ // Round
  end
  if $\exists i : \delta \langle M_i, M_i^{\perp} \rangle^2 > \langle M_{i+1}, M_i^{\perp} \rangle^2 + \langle M_{i+1}, M_{i+1}^{\perp} \rangle^2$ **then**
    $M_i \leftrightarrow M_{i+1}$ // Swap
  end
  else return M // Done

Observe that the "Round" and "Swap" steps do not change the lattice. Next we show that the algorithm terminates within polynomial time, and, furthermore, that once it does, the basis is reduced.

Running time. The correctness of the algorithm, as well as the fact that it runs in polynomial time, are proved simultaneously by introducing a parameter. This parameter is at most exponential in the dimension n at the start of the execution, and at each step is reduced by a constant factor < 1, while never going below the determinant multiplied by the minimal quanta (precision) of the vectors.

Fact B.25. *Given an integer basis M, consider*

$$\mathcal{DD}[M] \stackrel{\text{def}}{=} \prod_{i=1}^{n} \left| \langle M_i, M_i^{\perp} \rangle \right|^{n-i+1}$$

which satisfies the following properties:

- $\mathcal{DD}[M] \leq (\max_{1 \leq i \leq n} \|M_i\|)^{\frac{n(n+1)}{2}}$;
- $\mathcal{DD}[M] \geq 1$;
- $\mathcal{DD}[M]$ *is reduced by a factor* $\sqrt{\delta}$ *in each swap step of the LLL algorithm, and does not increase by an orthogonalization step.*

B.6. Extension

This algorithm can be improved to approximate CVP to within same ratio, and SVP to within $2^{n \frac{\log \log n}{\log n}}$.

B.7. CVP algorithm

Covering radius. Let us define yet another interesting parameter of a lattice. How far can a vector be from the lattice $\mathcal{L}$ (that is, the distance from its closest lattice vector)?

Definition B.15 (Covering tadius).

$$\mu(\mathcal{L}) \stackrel{\text{def}}{=} \max_{t \in \mathbb{R}^n} \operatorname{dist}(t, \mathcal{L}) = \inf\left\{ r > 0 \mid \mathbb{R}^n \subset \bigcup_{v \in \mathcal{L}} \mathcal{B}(v, r) \right\}.$$

Claim B.26.

$$\frac{1}{2}\lambda_n[\mathcal{L}] \leq \mu(\mathcal{L}) \leq \frac{1}{2}\sqrt{n}\lambda_n[\mathcal{L}].$$

B.7.1. Babai's nearest plane algorithm

Let us now present an algorithm for approximating CVP à la LLL [16]. Let $M_1, \ldots, M_n$ be the LLL-reduced basis for $\mathcal{L}$. Every vector $v \in \mathcal{L}$ can be represented as Mz for $z \in \mathbb{Z}^n$. Let $H_k \stackrel{\text{def}}{=} \{Mz \mid z_n = k\}$ for $k \in \mathbb{Z}$ which altogether partition all lattice vectors in $\mathcal{L}$, each belonging to some hyperplane H_k. Note that, to find an approximately closest lattice vector to t, one could find the closest hyperplane H_k to t; then, move t so it is at the same relation with H_0 as it was with H_k and continue recursively on the $(n-1)$-dimensional sublattice.

Algorithm 2: Babai's algorithm

Input: full-rank LLL-reduced matrix $M, t \in \mathbb{R}^n$
Output: A lattice vector v relatively close to t
$v = \vec{0}, t' = t$
for $n' = n$ *to* 1 **do**
  $k = \lfloor \langle t', M_{n'}^{\perp} \rangle \rceil$ `// Find nearest hyperplane`
  $v = v + kM_{n'}$ `// Remember to add` $kM_{n'}$ `back`
  $t' = t' - kM_{n'}$ `// Continue w/ sublattice where coefficient of` $M_{n'}$ `is 0`
end
return v

Analysis. First, observe that the algorithm runs in a polynomial time and returns a lattice vector v. One observes that

(1) $t' + v = t$ always.

(2) After the stage in the loop when $n' = j$, it holds that

$$\left|\langle t', M_j^{\perp} \rangle\right| \leq \frac{1}{2}\left|\langle M_j, M_j^{\perp} \rangle\right|,$$

and that continues to be true throughout (as both sides never change).

Therefore, at the end of the run

$$\left\|t'\right\| \leq \frac{1}{2}\sqrt{\sum_{i=1}^{n}\left\langle M_i, M_i^{\perp}\right\rangle^2}.$$

Corollary B.27.

$$\mu\left(\mathcal{L}[M]\right) \leq \frac{1}{2}\sqrt{\sum_{i=1}^{n}\left\langle M_i, M_i^{\perp}\right\rangle^2}.$$

Claim B.28. *Running Babai's algorithm on $\frac{3}{4}$-LLL (where $\delta = \frac{3}{4}$) gives an error of*

$$\left\|t'\right\| \leq 2^{\frac{n}{2}-1}\left|\left\langle M_n, M_n^{\perp}\right\rangle\right|.$$

Corollary B.29. *Running Babai's algorithm on $\frac{3}{4}$-LLL is $2^{\frac{n}{2}}$-approximation algorithm for CVP.*

REFERENCES

[1] L. M. Adleman, Factoring and lattice reduction, draft March 16, 1995.

[2] D. Aharonov and O. Regev, Lattice problems in NP ∩ coNP. *J. ACM* **52** (2005), no. 5, 749–765.

[3] R. Ahlswede and L. H. Khachatrian, The complete intersection theorem for systems of finite sets. *Eur. J. Comb.* **18** (1997), no. 2, 125–136.

[4] M. Ajtai, Generating hard instances of lattice problems. In *Proc. 28th ACM Symposium on the Theory of Computing (Philadelphia, Pennsylvania, USA)*, pp. 99–108, ACM, New York, NY, USA, 1996.

[5] M. Ajtai, The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proc. 30th ACM Symposium on the Theory of Computing (Dallas, Texas, USA)*, pp. 10–19, ACM, New York, NY, USA, 1998.

[6] *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25–28, 2008, Philadelphia, PA, USA*, IEEE Computer Society, 2008.

[7] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.* **54** (1997), no. 2, 317–331.

[8] S. Arora, B. Barak, and D. Steurer, Subexponential algorithms for unique games and related problems. *J. ACM* **62** (2015), no. 5, 42:1–42:25.

[9] S. Arora, E. Berger, E. Hazan, G. Kindler, and M. Safra, On non-approximability for quadratic programs. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23–25 October 2005, Pittsburgh, PA, USA, Proceedings*, pp. 206–215, IEEE Computer Society, 2005.

[10] S. Arora, S. Khot, A. Kolla, D. Steurer, M. Tulsiani, and N. K. Vishnoi, Unique games on expanding constraint graphs are easy: extended abstract. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008*, pp. 21–28, ACM, 2008.

[11] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and the hardness of approximation problems. *J. ACM* **45** (1998), no. 3, 501–555.

[12] S. Arora and S. Safra, Probabilistic checking of proofs: A new characterization of NP. *J. ACM* **45** (1998), no. 1, 70–122.

[13] P. Austrin, Balanced max 2-SAT might not be the hardest. In *Proc. 39th ACM Symposium on the Theory of Computing (San Diego, California, USA)*, pp. 189–197, ACM, New York, NY, USA, 2007.

[14] P. Austrin, S. Khot, and M. Safra, Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory of Computing* **7** (2011), no. 3, 27–43.

[15] L. Babai, Trading group theory for randomness. In *Proc. 17th ACM Symposium on Theory of Computing (Providence, Rhode Island, USA)*, pp. 421–429, ACM, New York, NY, USA, 1985.

[16] L. Babai, On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica* **6** (1986), no. 1, 1–13.

[17] B. Barak, F. G. S. L. Brandão, A. W. Harrow, J. A. Kelner, D. Steurer, and Y. Zhou, Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19–22, 2012*, pp. 307–326 ACM, 2012.

[18] B. Barak, P. K. Kothari, and D. Steurer, Small-set expansion in shortcode graph and the 2-to-2 conjecture. *CoRR* (2018), arXiv:1804.08662.

[19] M. Bellare, O. Goldreich, and M. Sudan, Free bits, PCPs, and nonapproximability – towards tight results. *SIAM Journal on Computing* **27** (1998), no. 3, 804–915.

[20] Z. Brakerski and V. Vaikuntanathan, Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing* **43** (2014), no. 2, 831–871.

[21] M. Charikar, K. Makarychev, and Y. Makaryshev, Near-optimal algorithms for unique games. In *Proc. 38th ACM Symposium on Theory of Computing*, pp. 205–214, ACM, New York, NY, USA, 2006.

[22] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar, On the hardness of approximating multicut and sparsest-cut. *Computational Complexity* **15** (2006), no. 2, 94–114.

[23] S. A. Cook, The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC'71*, pp. 151–158, ACM, New York, NY, USA, 1971.

[24] N. R. Devanur, S. Khot, R. Saket, and N. K. Vishnoi, Integrality gaps for sparsest cut and minimum linear arrangement problems. In *Kleinberg* [58], pp. 537–546.

[25] I. Diakonikolas, D. Kempe, and M. Henzinger (eds.), In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25–29, 2018*, ACM, 2018.

[26] I. Dinur, The PCP theorem by gap amplification. In *Kleinberg* [58], pp. 241–250.

[27] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra, PCP characterizations of NP: toward a polynomially-small error-probability. *Comput. Complex.* **20** (2011), no. 3, 413–504.

[28] I. Dinur, S. Khot, G. Kindler, D. Minzer, and M. Safra, On non-optimally expanding sets in Grassmann graphs. In *Diakonikolas et al.* [25], pp. 940–951.

[29] I. Dinur, S. Khot, G. Kindler, D. Minzer, and M. Safra, Towards a proof of the 2-to-1 games conjecture? In *Diakonikolas et al.* [25], pp. 376–389.

[30] I. Dinur, G. Kindler, and S. Safra, Approximating-CVP to within almost-polynomial factors is NP-hard. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pp. 99–109, IEEE, 1998.

[31] I. Dinur and S. Safra, The importance of being biased. In *Proc. 34th Annual ACM Symposium on Theory of Computing*, pp. 33–42, ACM, New York, NY, USA, 2002.

[32] I. Dinur and S. Safra, On the hardness of approximating minimum vertex cover. *Ann. of Math. (2)* **162** (2005), no. 1, 439–485.

[33] L. Eldar and S. Hallgren, An efficient quantum algorithm for lattice problems achieving subexponential approximation factor, 2022.

[34] P. Erdős, C. Ko, and R. Rado, Intersection theorems for systems of finite sets. *The Quarterly Journal of Mathematics* **12** (1961), no. 1, 313–320.

[35] U. Feige, A threshold of $\ln n$ for approximating set cover. *J. ACM* **45** (1998), no. 4, 634–652.

[36] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy, Interactive proofs and the hardness of approximating cliques. *Journal of the ACM* **43** (1996), no. 2, 268–292.

[37] E. Friedgut, Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica* **18** (1998), no. 1, 27–35.

[38] M. X. Goemans and D. P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* **42** (1995), no. 6, 1115–1145.

[39] O. Goldreich and S. Goldwasser, On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences* **60** (2000), no. 3, 540–563.

[40] A. Gupta and K. Talwar, Approximating unique games. In *Proceedings of the Seventeenth Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22–26, 2006*, pp. 22–26, ACM Press, 2006.

[41] V. Guruswami, R. Manokaran, and P. Raghavendra, Beating the random ordering is hard: inapproximability of maximum acyclic subgraph. In *Proc. Annual IEEE Symposium on Foundations of Computer Science* [6], pp. 573–582.

[42] J. Håstad, Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Math.* **182** (1999), no. 1, 105–142.

[43] J. Håstad, Some optimal inapproximability results. *J. ACM* **48** (2001), no. 4, 798–859.

[44] D. Hilbert and W. Ackermann, *Grundzüge der theoretischen Logik*. Grundlehren der mathematischen Wissenschaften 27, Springer, Berlin Heidelberg, 1938.

[45] R. M. Karp, Reducibility among combinatorial problems. In *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*, pp. 85–103, Springer US, Boston, MA, 1972.

[46] S. Khot, Improved inaproximability results for MaxClique, chromatic number and approximate graph coloring. In *Proc. 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14–17 October 2001, Las Vegas, Nevada, USA*, pp. 600–609, IEEE Computer Society, NW Washington, DC, USA, 2001.

[47] S. Khot, On the power of unique 2-prover 1-round games. In *Proceedings of the 17th IEEE Annual Conference on Computational Complexity, Montréal, Québec, Canada, May 21–24, 2002*, IEEE Computer Society, NW Washington, DC, USA, p. 25, 2002.

[48] S. Khot, Hardness of approximating the shortest vector problem in high L_p norms. In *Proc. 44th IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society, NW Washington, DC, USA, p. 290, 2003.

[49] S. Khot, Hardness of approximating the shortest vector problem in lattices. *J. ACM* **52** (2005), no. 5, 789–808.

[50] S. Khot, Inapproximability of NP-complete problems, discrete fourier analysis, and geometry. In *Proceedings of the International Congress of Mathematicians 2010*, pp. 2676–2697, Hindustan Book Agency, New Delhi, 2010.

[51] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell, Optimal inapproximability results for MAX-CUT and other 2-variable CSPS? *SIAM J. Comput.* **37** (2007), no. 1, 319–357.

[52] S. Khot, D. Minzer, and M. Safra, On independent sets, 2-to-2 games, and Grassmann graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19–23, 2017*, pp. 576–589, ACM, New York, NY, USA, 2017.

[53] S. Khot, D. Minzer, and M. Safra, Pseudorandom sets in Grassmann graph have near-perfect expansion. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 592–601, IEEE, 2018.

[54] S. Khot and R. O'Donnell, SDP gaps and UGC-hardness for Max-Cut-Gain. *Theory of Computing* **5** (2009), no. 1, 83–117.

[55] S. Khot and O. Regev, Vertex cover might be hard to approximate to within $2-\varepsilon$. *J. Comput. Syst. Sci.* **74** (2008), no. 3, 335–349.

[56] S. Khot and N. K. Vishnoi, The unique games conjecture, integrality gap for cut problems and embeddability of negative-type metrics into l_1. *J. ACM* **62** (2015), no. 1, 8:1–8:39.

[57] G. Kindler, A. Naor, and G. Schechtman, The ugc hardness threshold of the l_p Grothendieck problem. In *SODA*, edited by S.-H. Teng, pp. 64–73, SIAM, 2008.

[58] J. M. Kleinberg (ed.), In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21–23, 2006*, ACM, 2006.

[59] A. Kolla, Spectral algorithms for unique games. *Computational Complexity* **20** (2011), no. 2, 177–206.

[60] A. Korkine and G. Zolotareff, Sur les formes quadratiques. *Math. Ann.* **6** (1873), no. 3, 366–389.

[61] A. K. Lenstra, H. W. Lenstra, and L. Lovász, Factoring polynomials with rational coefficients. *Math. Ann.* **261** (1982), 513–534.

[62] L. Levin, Universal search problems. *Probl. Peredachi Inf.* **9** (1973), no. 3, 115–116.

[63] V. Lyubashevsky, Fiat-Shamir with aborts: applications to lattice and factoring-based signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 598–616, Springer, 2009.

[64] P. Manurangsi, Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. *Algorithms* **11** (2018), no. 1, 10.

[65] D. Micciancio, The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing* **30** (2001), no. 6, 2008–2035. Preliminary version in FOCS 1998.

[66] D. Micciancio and O. Regev, Worst-case to average-case reductions based on Gaussian measures. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17–19 October 2004, Rome, Italy, Proceedings*, pp. 372–381, IEEE Computer Society, 2004.

[67] H. Minkowski, *Geometrie der Zahlen*. B.G. Teubner, 1896.

[68] H. Minkowski, *Geometrie der Zahlen* 40, Teubner, 1910.

[69] D. Moshkovitz and R. Raz, Sub-constant error probabilistically checkable proof of almost-linear size. *Comput. Complex.* **19** (2010), no. 3, 367–422.

[70] E. Mossel, Gaussian bounds for noise correlation of functions and tight analysis of long codes. In *Proc. Annual IEEE Symposium on Foundations of Computer Science* [6], pp. 156–165.

[71] E. Mossel, R. O'Donnell, and K. Oleszkiewicz, Noise stability of functions with low influences: invariance and optimality. *Annals of Mathematics* (2010), 295–341.

[72] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, 1982.

[73] P. Raghavendra, Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC'08*, pp. 245–254, ACM, New York, NY, USA, 2008.

[74] P. Raghavendra and D. Steurer, Graph expansion and the unique games conjecture. In *Proc. 42nd ACM Symposium on Theory of Computing*, pp. 755–764, ACM, New York, NY, USA, 2010.

[75] P. Raghavendra, D. Steurer, and M. Tulsiani, Reductions between expansion problems. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26–29, 2012*, pp. 64–73, IEEE Computer Society, 2012.

[76] R. Raz, A parallel repetition theorem. *SIAM J. Comput.* **27** (1998), no. 3, 763–803.

[77] O. Regev, On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **56** (2009), no. 6.

[78] O. Regev and N. Stephens-Davidowitz, A reverse Minkowski theorem. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19–23, 2017*, edited by H. Hatami, P. McKenzie, and V. King, pp. 941–953, ACM, 2017.

[79] C. P. Schnorr, A hierarchy of polynomial-time basis reduction algorithms. *Theoretical Computer Science* **53** (1987), no. 2–3, 201–224.

[80] U. Shapira and B. Weiss, Stable lattices and the diagonal group. *Journal of the European Mathematical Society* **18** (2016), no. 8, 1753–1767.

[81] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, IEEE Computer Society, NW Washington, DC, USA, 1994.

[82] E. M. Stein and G. Weiss, *Introduction to Fourier analysis on Euclidean spaces*. Princeton Mathematical Series 32, 1971.

[83] L. Trevisan, Approximation algorithms for unique games. *Theory of Computing* **4** (2008), no. 1, 111–128.

[84] W. Banaszczyk, New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen* 296 (1993), no. 1. 625–635.

MULI (SHMUEL) SAFRA

Quantum Science and Technology, Tel Aviv University, P.O. Box 39040,
Tel Aviv 6997801, Israel, safra@tauex.tau.ac.il