The non-injective hidden shift problem

by

Mirmojtaba Gharibi

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2011

Abstract

In this work, we mostly concentrate on the hidden shift problem for non-injective functions. It is worthwhile to know that the query complexity of the non-injective hidden shift problem is exponential in the worst case by the well known bounds on the unstructured search problem. Hence, we can make this problem more tractable by imposing additional constraints on the problem. Perhaps the first constraint that comes to mind is to address the average case problem. In this work, we show that the average case non-injective hidden shift problem can be reduced to the injective hidden shift problem by giving one such reduction. The reduction is based on a tool we developed called *injectivization*. The result is strong in the sense that the underlying group can be any finite group and that the non-injective functions for which we have defined the hidden shift problem can have range in an arbitrary finite set. Using this tool, we simplify the main result of a recent paper by Gavinsky, Roetteler, and Roland [2011] about the hidden shift problem for Boolean-valued functions by reducing that problem to Simon's problem. They also posed an open question which is subject to personal interpretation. We answer the seemingly most general interpretation of the question. However, we use our own techniques in doing so (the authors ask if their techniques can be used for addressing that problem). Another constraint that one can consider is to have a promise on the structure of the functions. In this work we consider the hidden shift problem for *c-almost generalized bent* functions. A class of functions which we defined that includes the generalized bent functions. Then we turn our attention toward the generalized hidden shift problem which is easier than injective hidden shift problem and hence more tractable. We state some of our observations about this problem. Finally we show that the average classical query complexity of the non-injective hidden shift problem over groups of form $\mathbb{Z}_m^n$ when m is a constant is exponential, which also immediately implies that the classical average query complexity of the non-injective hidden shift problem is exponential. We also show that the worst-case classical query complexity of the generalized injective hidden shift problem over the same group is high, which implies that the classical query complexity of the hidden shift problem is high.

Acknowledgements

I would like to express my deep gratitude to my supervisor, Richard Cleve, for continuous support, many fruitful discussions, and his comments on the draft of the thesis. I also like to thank Andrew Childs for extremely helpful discussions during the earlier stage of my work. I would like to thank my father, Hossein Gharibi, for his patience and giving helpful directions on my research. Special thanks to John Watrous for his time, patience, and advices. I also thank Dmitry Gavinsky for helpful email correspondence.

I especially thank my friend and beloved, my wife. Zahra, this was long process and I appreciate your unconditional support and the love that you gave me through it all . I clearly see how your hands and heart helped me in this long journey.

Dedication

This thesis is dedicated to Zahra, my wife; being with her has been the most life changing experience for me. This is dedicated to her, for all the courage she gave to me for pursuing my dreams.

This is also dedicated to my father and my mother who gave me my highest values by being a constant source of inspiration to me. Living with them was the true everlasting gift of life to me.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Why the hidden shift problem?

The first formal evidence that a quantum computer might be stronger than a classical computer was given in the work of Bernstein and Vazirani [1993]. In that paper, they defined the complexity class BQP, the class of problems that can be solved efficiently on a quantum computer. They showed that BQP contains BPP. They also introduced a problem called Recursive Fourier Sampling to demonstrate a super-polynomial separation between BQP and BPP in the relativized world. Subsequently, Simon came up with a black-box problem which could be solved exponentially faster on a quantum computer [Simon, 1994]. Although the problem was of little practical interest, it was a milestone in the history of quantum computing. Firstly, it was the first problem for which an exponential separation could be achieved, giving even stronger evidence for the superior power of quantum computers. Secondly, the technique used in solving the problem paved the road for the development of an efficient quantum algorithm for two long-standing open problems in the history of mathematics; factoring and discrete log problems. Inspired by his work, Shor discovered an efficient quantum algorithm for solving these problems on a quantum computer. This brought in an era of excitement about the quantum computer. The research both on the quantum algorithms and on practical realization of a quantum computer received a major boost from this discovery [Shor, 1997]. It was later shown that Simon's problem, factoring and discrete log problem all can be cast as special cases of the abelian hidden subgroup problem. Now, we know that hidden subgroup problem over abelian groups can be solved efficiently on a quantum computer. In the light of these results, a whole body of research is devoted to finding algorithms for the hidden subgroup problem over non-abelian groups. However, these efforts have met with only limited success. In particular, we do not know yet of any quantum algorithm that efficiently solves the hidden subgroup problem over the *dihedral* group and *symmetric group*. The hidden subgroup problem over these groups is particularly interesting due to their connection to some lattice problems and graph isomorphism.

The slow progress in finding new algorithms for the hidden subgroup problem has motivated researchers to think of new paradigmatic problems in which a quantum computer can demonstrate a superior performance. One notable example is the hidden shift problem[van Dam et al., 2003][Friedl et al., 2003]. The study of the hidden shift problem has helped us in giving insights into the hidden subgroup problem due to some connections between the two. In particular, we know that the injective hidden shift problem with an underlying abelian group is equivalent to an instance of the non-abelian hidden subgroup problem. As one example, the hidden shift problem over the group $\mathbb{Z}_N$ is equivalent to the hidden subgroup problem over the *dihedral* group. Some problems of interest can be cast both as a hidden shift problem and as a hidden subgroup problem. Most notably, graph isomorphism is one such problem. It is even argued in [Childs and Wocjan, 2007] that viewing graph isomorphism as a hidden shift problem is a more natural approach than viewing it as a hidden subgroup problem. Beside all these, some instances of the hidden shift problem are interesting on their own.

Usually in both the hidden shift problem and hidden subgroup problem we rely on the ability of quantum computers in performing the quantum Fourier transform. Often, the difference is that in the hidden shift problem we use the relation between the Fourier transform of shifted functions whereas in the hidden subgroup problem, we rely on the ability of the Fourier transform to extract the periodicity.

1.1.1 Connections to the hidden subgroup problem

There is a well-known connection between the hidden shift problem and the hidden subgroup problem. More specifically, the hidden shift problem defined over an abelian group G is equivalent to the hidden subgroup problem defined over the group of the semi-direct product of G and $\mathbb{Z}_2$ where $\mathbb{Z}_2$ acts on G by inversion[Childs and Wocjan, 2007].

In addition to this direct connection, many times we see a problem that admits reduction both to a hidden shift problem and a hidden subgroup problem. This establishes an indirect connection between the hidden shift problem and the hidden subgroup problem as the tools enabling us to look at the same problem from different perspectives.

Perhaps the most interesting problem in this category is graph isomorphism. Since there is no known reduction from every NP problem to this problem, it is not known to be an NP-complete problem together with factoring and discrete log problem. However, this problem is in NP since the validity of an isomorphism can be quickly verified. Unlike factoring and discrete log, no polynomial time quantum algorithm is known for solving this problem. Graph isomorphism can be reduced to the hidden subgroup problem [Beals, 1997][Ettinger and Høyer, 1999a][Høyer, 1997][Boneh and Lipton, 1995] or to a hidden shift problem both over the symmetric group. Considering the hidden subgroup problem over S_{2n} or $S_n \wr \mathbb{Z}_2$, there is a reduction from graph isomorphism to the first problem where the hidden subgroups are formed by full support involutions and to the second one with the hidden subgroups formed by involutive swaps.

One can also consider the hidden shift problem over S_n for studying the graph isomorphism. This problem can be reduced to both of the hidden subgroup problems above. In other words, this problem might be easier than the problems above (when viewed as black-box problems), but it is definitely no harder. Furthermore, when viewing graph isomorphism as a hidden shift problem, every possible shift corresponds to a possible isomorphism. However, not every possible hidden subgroup in the problems above corresponds to a possible hidden shift [Childs and Wocjan, 2007].

1.2 Preliminaries

1.2.1 Notations and basic definitions

- We use G to refer to an arbitrary finite group and S to refer to an arbitrary finite set. Since we are dealing with abelian groups in all of this thesis, except of chapter 3 where we address the general groups, with a slight abuse of notation, we denote the groups additively. There is an arbitrary choice of left or right addition in the non-abelian hidden shift problem. Without loss of generality, we choose right addition. All groups in this thesis are additively denoted.

- For an m-tuple $V \in G^m$ we denote its component k as v_k.

- For any function $f : \mathbb{Z}_m^n \to \mathbb{Z}_m$, we show with capital letter, the corresponding function $F : \mathbb{Z}_m^n \to \{\omega_m^x | x \in \mathbb{Z}_m\}$ defined by $F(x) = \omega_m^{f(x)}$.

- For any discrete function, we denote the size of its domain with N.

- For any function $g : \mathbb{Z}_m^n \to \mathbb{C}$ we define its Fourier transform $\hat{g} : \mathbb{Z}_m^n \to \mathbb{C}$ by

$$\hat{g}(s) = \frac{1}{N} \sum_{x \in \mathbb{Z}_m^n} \omega_m^{x \cdot s} g(x)$$

  where ω_m is the m'th root of unity and $x \cdot s$ is the inner product of x and s modulo m.

1.2.2 Highly non-linear functions

A function $f : \mathbb{Z}_2^n \to \{0,1\}$ is called bent if each of the Fourier coefficients of F is either $\frac{+1}{\sqrt{N}}$ or $\frac{-1}{\sqrt{N}}$, i.e. for all $z \in \mathbb{Z}_2^n$:

$$|\hat{F}(z)|^2 = \frac{1}{N}.$$

The notion of bent function is first defined in [Rothaus, 1976]. Bent functions are of particular interest in cryptography. Their special properties make them resistant to specific type of attacks. Bent functions are sometimes called maximum non-linear functions. This terminology is motivated by the fact that by comparing the values that a bent function and set of all linear functions take at each point, we see that their disagreement is maximal. This rules out the possibility of successful attacks aimed at extracting dependencies between the values the bent function takes and some linear subspace of the arguments [Rothaus, 1976].

It is not hard to prove that bent functions do not exist for values of n that are odd. For large enough values of n that are even, bent functions are guaranteed to exist. In particular, some classes of bent functions are well characterized. However, a complete characterization of bent functions seems to be a subtle task. One of the biggest known classes of bent functions are called Maiorana-Mcfarland bent functions. The number of bent functions in this class is $(2^{n/2})! \times 2^{2^{n/2}}$. This number is asymptotically equivalent to $\left(\frac{2^{n/2+1}}{e}\right)^{2^{n/2}} \sqrt{2\pi 2^{n/2}}$ which can be shown by using Stirling's formula. However, still it seems that the number of bent functions in this class is negligible compared to the total number of bent functions. The exact count of the number of bent functions is not known. By Rothaus inequality we know any bent function has a degree of at most $n/2$. The naive bound that results from this constraint suggests that the total count of bent functions is at most $2^{\left(2^{n-1}+1/2\binom{n}{n/2}\right)}$ [Carlet and Gaborit, 2006]. A better bound was recently discovered in [Carlet and Klapper, 2002], but it still seems to be far from being tight [Carlet and

Gaborit, 2006]. For $n = 6$ we know the exact count of bent function which is almost equal to 2^{32} [Preneel, 1993].

The most common generalization of bent functions can be seen in [Kumar et al., 1985]. The generalized bent function is defined on the set of n-tuples with each component being a member of $\mathbb{Z}_p$. Here we consider the case where these n-tuples correspond to the members of the group $\mathbb{Z}_p^n$. Any function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ is called bent, if $\left|\hat{F}(z)\right|$ is constant over all $z \in \mathbb{Z}_p^n$ [Kumar et al., 1985]. Interestingly, generalized bent functions and bent functions share many properties that are useful in cryptography. Often, in applications, only the case where p is a prime number is considered. It is noteworthy to mention that the quantum algorithms we present in this thesis for solving the hidden shift problem for generalized bent functions cover the case when p is a constant prime power. Hence, we have covered the case that is of most practical interest (the case of prime p) in this thesis. Unlike the case of bent functions which do not exist when n is an odd number, generalized bent functions exist regardless of the parity of n when p is an odd prime number [Nyberg, 1991].

In addition to generalized bent functions, we define a set of functions which include the generalized bent functions and for which we can solve the hidden shift problem using the quantum algorithm presented in this thesis. We say a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ is a c-almost generalized bent function if the absolute value of the ratio of the smallest Fourier coefficient of F to its biggest Fourier coefficient is the non-zero constant c. More formally

$$\frac{\min_x \left|\hat{F}(x)\right|}{\max_y \left|\hat{F}(y)\right|} = c$$

where $c > 0$ is a constant.

1.3 Hidden Shift Problem

In the injective hidden shift problem, we are given two injective functions over some group G that are simply a shifted version of each other. The task is to output such a shift. More formally:

Definition (Hidden Shift Problem). Let $f, g : G \to S$ be two injective functions given by blackboxes such that, for some unique $s \in G$, $f(x) = g(x + s)$ for all $x \in G$. The goal is to find the *hidden shift* s.

Since we deal with the general groups only in chapter 3 and except of that we work only with abelian groups, with a slight abuse of notation, we denote the groups additively, even when the groups are non-abelian.

Interesting problems can be stated as or reduced to a hidden shift problem. Most notably, these include hidden subgroup problem over dihedral group which, as noted earlier, is equivalent to the hidden shift problem over $\mathbb{Z}_N$[Ettinger and Høyer, 1999b][Kuperberg, 2005] and graph isomorphism, which can be reduced to the hidden shift problem over S_n[Childs and Wocjan, 2007].

It is possible to define a variation of the hidden shift problem by relaxing the requirement for the functions to be injective. However, then a self-shift in the functions might be possible which causes s to not be unique. We say there is a self-shift in the functions if for some $t \in G \setminus 0$ we have $f(x+t) = f(x)$ which also implies $g(x+t) = g(x)$ for all x). To make the problem well defined, we make a promise about the function f not to be self-shift.

In this thesis, we are mostly interested in the non-injective hidden shift problem. This relaxed version of the hidden shift problem has also attracted scholars in the field.

1.4 Previous work

1.4.1 Non-injective hidden shift problem

In addition to the injective hidden shift problem, the non-injective version of the hidden shift problem where functions are not required to be injective has been studied. An efficient quantum algorithm is given in [Rötteler, 2010] for solving the hidden shift problem for several classes of the so-called bent functions. Gavinsky et al. [2011] gave an efficient quantum algorithm for solving the hidden shift problem for average case Boolean functions $f : \mathbb{Z}_2^n \to \{0, 1\}$. They could show that their algorithm addresses the hidden shift problem for all bent functions as well. Ozols et al. [2011] gave another algorithm for the Boolean hidden shift problem based on the Quantum reject sampling. An efficient quantum algorithm for solving the hidden shift problem when $f : \mathbb{Z}_p \to \{-1, 0, 1\}$ is the Legendre symbol is presented in the work by van Dam et al. [2003] who for the first time explicitly defined the hidden shift problem. They also gave a reduction to the hidden shift problem over $\mathbb{Z}_p$ assuming a conjecture in [Boneh and Lipton, 1995] holds. The conjecture states that any string formed by l subsequent values of f is unique where $l > 2\log^2 p$.

By lower bounds on the query complexity of the unstructured search problem[Bennett et al., 1997], a worst case solution to the non-injective hidden shift problem cannot be obtained. Imposing restrictions on the instances, makes the non-injective hidden shift problem more tractable. Perhaps, one of the most common restriction in this situation is to consider the average case problem.

Since we have devoted some pages in this thesis to simplifying the main result in [Gavinsky et al., 2011] and to answering seemingly the most general interpretation of the open question posed there, it is appropriate to explain their result and approach in somewhat more detail. We devote the next subsection to this.

1.4.2 The hidden shift problem for Boolean functions

As mentioned in the previous subsection, Gavinsky, Roetteler, and Roland [2011] in a recent paper defined a hidden shift problem for Boolean functions called the Boolean hidden shift problem and could show an exponential separation for the average case of the problem. In this subsection, we describe their work in some details.

In the Boolean hidden shift problem, we are given two functions $f, g : \mathbb{Z}_2^n \to \{0,1\}$ as oracles. Firstly, it holds for some $v \in \mathbb{Z}_2^n$ that $f(x) = g(x+s)$ for all x in the domain. Secondly, it holds that if $f(x) = f(x+t)$ for all x in the domain, then $t = 0$. In other words, the functions do not have a self-shift. The goal is to find v.

The familiar notion of *influence* of η over f is defined as

$$\gamma_{f,\eta} \triangleq \Pr_x\{f(x) \neq f(x+\eta)\}$$

where x is chosen uniformly at random. In some sense, $\gamma_{f,\eta}$ captures the extent to which our function f is a self-shift function with respect to η. The *minimum influence* of f is defined as the minimum of the above quantity over all η.

The idea for solving an average case of this problem is to prepare a set of n linearly independent equations of components of s which uniquely determine s. The expected number of queries and time steps are $O(n)$ and $O(poly(n))$ as shown in [Gavinsky et al., 2011]. This result is obtained by relating the amount of queries and time needed to solve any instance of the Boolean hidden shift problem to the minimum influence of the function f. While they show in [Gavinsky et al., 2011] that the average case query complexity of the Boolean hidden shift problem is high, they give an efficient quantum algorithm to address the average case. In their sampling subroutine, they measure the states of form

$$\sum_{u\in\mathbb{Z}_2^n}\frac{\hat{F}(u)(1+(-1)^{u.s})}{2}|0\rangle|u\rangle+\sum_{u\in\mathbb{Z}_2^n}\frac{\hat{F}(u)(1-(-1)^{u.s})}{2}|1\rangle|u\rangle.$$

to obtain equations of form $u\cdot s=b$ for some known random value of u and some known value for b consistent with s.

As an open question, they ask "whether these methods can be generalized and adapted for the case of non-Boolean functions also". It seems that the question they pose is subject to personal interpretation. We explain how we approach this question in the next section.

1.5 Summary of results

In this thesis, we are mostly concerned with the average case non-injective hidden shift problem. We introduce a framework that we call *injectivization* for reducing the average case non-injective hidden shift problem to the injective hidden shift problem. This is a strong tool, since, firstly, the underlying group can be an arbitrary group and we do not impose any restriction on the structure of the group. Secondly, the functions involved can have arbitrary range. As a specific application of this tool, we use it to reduce the average case non-injective hidden shift problem when the underlying group is of form $\mathbb{Z}_q^n$ for q a prime power to its injective counterpart. When $q=2$, Using this framework, we simplify the main results in [Gavinsky et al., 2011] by reducing the Boolean hidden shift problem to the well known Simon's problem. As an open question in [Gavinsky et al., 2011], they ask if their methods in solving the Boolean hidden shift problem (over group $\mathbb{Z}_2^n$) can be generalized for solving the hidden shift problem for non-Boolean functions. Unfortunately, this open question is subject to personal interpretation. We first make it clear that we have not used the method in [Gavinsky et al., 2011] to answer their open question. One way for generalizing the Boolean hidden shift problem is to keep the group unchanged but let the functions in the Boolean hidden shift problem also take values other than $\{0,1\}$. We show that, by using our injectivization method, this problem can again be reduced to Simon's problem. Another generalization that one may consider is also to allow the group to be a similar cyclic group such as $\mathbb{Z}_q^n$ and allow the function to range in any arbitrary set. We can address this problem too using injectivization when q is a constant prime power. This is because the injective hidden shift problem over those groups can be reduced to solving a set of polynomial equations, which is already addressed in [Friedl et al., 2003] and [Ivanyos, 2008]. We give a short survey of those results in chapter 2. Unfortunately, when q is a non-prime power compostie number, no algorithm is known to solve the injective hidden shift problem efficiently. This is our main motive for investigating the complexity of this problem by defining an easier problem in chapter 5. In chapter 3, we present our injectivization framework.

As mentioned, in chapter 3, we give a solution for the average case instances of the non-injective hidden shift problem over groups of form $\mathbb{Z}_q^n$ when q is a constant prime power. However, we do not know if this can be used to solve the hidden shift problem for certain functions of interest. Among these functions, one can point out the generalized bent functions. In chapter 4, we generalize the algorithm in [Gavinsky et al., 2011] and then use the results in [Ivanyos, 2008] to give a solution for the hidden shift problem for the class of c-almost generalized bent functions (which includes the generalized bent functions) when the underlying group is $\mathbb{Z}_q^n$ for q a constant prime power.

Childs and van Dam [2007] have defined the generalized hidden shift problem for the group $\mathbb{Z}_N$ in the spirit of finding a problem that is harder than abelian HSP and still easier than the hidden shift problem for a quantum computer but intractable for classical computers. Basically, in that problem, they allow the computers to access more information by providing them with extra shifted oracles. In chapter 5, we extend this definition for the hidden shift problem over cyclic groups of form $\mathbb{Z}_m^n$. Our main motive is to determine the minimum number of oracles we need to address the hidden shift problem when m is a non-prime power composite number by using currently known methods. This is because the hidden shift problem with m, being a non-prime power composite number, has been open for almost a decade now.

In chapter 6, we prove that, to classically solve the average case non-injective hidden shift problem over groups $\mathbb{Z}_m^n$ and with functions having range in arbitrary finite set, one needs an exponential number of queries. Note that m is a constant number. We also prove that in order for a classical computer to solve the worst case instance of the injective generalized hidden shift problem, it will need an exponential number of queries, which immediately implies that the injective hidden shift problem over $\mathbb{Z}_m^n$ must also be hard.

In chapter 7, we describe some of the possible future research directions and offer some remarks and open questions.

Chapter 2

The hidden shift problem and the sets of equations and inequations

2.1 Linearization

Friedl et al. [2003] gave an efficient algorithm for solving the hidden shift problem over $\mathbb{Z}_p^n$ when p is a constant prime number. In this section we sketch the main idea they used to obtain this result. We also state the theorem that we will use from their work later.

They have a quantum sampling subroutine which at each iteration generates one inequation of form

$$u \cdot s \neq 0$$

by measuring a state of form

$$\sum_{u \in \mathbb{Z}_p^n} \omega^{u \cdot x} \cdot (1 - \omega_p^{u \cdot s}) \, |u\rangle$$

where $s \in \mathbb{Z}_p^n$ is the hidden shift and $u \in \mathbb{Z}_p^n$ is a random element that satisfies the above property and is drawn with probability at least c/p^n where c is some non-zero constant.

Then, using Fermat's little theorem, they raise both sides of the inequality to the power $p-1$ to obtain a set of polynomial equations of form

$$(u \cdot s)^{p-1} = 1$$

Unfortunately, the problem of solving polynomial equations over any finite field in general is NP-complete. However, the bright side is that this is not necessarily true for an overdefined set of equations. Hence, they use the idea of linearization to obtain a linear set of equations. This is done by thinking of each monomial in the above equation as an independent unknown variable. Hence, this will make the total number of unknown variables $\binom{n+p-1}{p-1}$ which is still a polynomial in n since p is a constant.

It is shown in their work that as u sweeps all the possible elements in $\mathbb{Z}_p^n$, the rank of the equations becomes complete. Then they show that, given any set of equation that does not have a full rank, with probability at least $1/2$, their quantum sampling subroutine outputs a linearly independent equation. Using these facts, they give an efficient quantum algorithm which they call *Translation Finding*$^f\left(\mathbb{Z}_p^n\right)$ for solving the hidden shift problem over $\mathbb{Z}_p^n$ whose complexity is stated in the theorem below.

Theorem 2.1. *For all positive integers n and prime numbers p, being given any instance of the hidden shift problem for functions $f, g : \mathbb{Z}_p^n \to S$, the quantum algorithm Translation Finding*$^f\left(\mathbb{Z}_p^n\right)$ *outputs the hidden shift with probability at least* $1/2$. *The query and time complexity of the algorithm a $O\left(p\left(n+p\right)^{p-1}\right)$ and $\left(n+p\right)^{O(p)}$ respectively.*

Using similar ideas, it is possible to give a quantum algorithm for solving the hidden shift problem over $\mathbb{Z}_{p^k}^n$ where p^k is a constant prime power. This problem is addressed in [Friedl et al., 2003] and [Ivanyos, 2008]. The idea is similar to above together with a usual encoding of elements of $\mathbb{Z}_{p^k}^n$ with elements in $\mathbb{Z}_p^{kn}$ by the expansion of each element in the $\mathbb{Z}_p$-linear basis. The algorithm in [Ivanyos, 2008] is based on a reduction to solving a set of random linear inequations. By measuring the states of form above, one can obtain a set of random linear inequations.

2.2 Set of random linear inequations

As it was seen in the previous section, solving the hidden shift problem over the group $\mathbb{Z}_q^n$ when q is a prime power was implicitly reduced to solving a set of polynomial equations in [Friedl et al., 2003]. Observing the fact that a promising approach to the hidden shift problem is solving a system of certain random inequations, Ivanyos [2008] formalizes such a framework and gives a quantum reduction from the hidden shift problem over any abelian group to solving a set of random linear inequations with special properties. He defines the following problem:

Definition (RANDOM LINEAR INEQUATIONS(A, c) - search version). Samples of the characters of a finite abelian group A are given, which are drawn according to a distribution that is nearly uniform with tolerance c on characters not containing a fixed element s in their kernels. The goal is to output the element $s \in A$.

A nearly uniform distribution with tolerance c is defined in the following way:

Definition 2.2. A distribution is said to be nearly uniform with a real tolerance $c \geq 1$ over a finite set D if there exists some subset of D called C such that $\Pr(x) = 0$ if and only if $x \in C$ and otherwise

$$\frac{1/c}{|D \setminus C|} \leq \Pr(x) \leq \frac{c}{|D \setminus C|}.$$

There is a quantum reduction from the hidden shift problem defined by $f, g : A \to S$ over abelian group A to the search version of RANDOM LINEAR INEQUATIONS(A, c). The reduction is specified below according to [Ivanyos, 2008]:

- Prepare three registers in the following initial state
$$|0\rangle|0\rangle|0\rangle$$
- Apply the quantum Fourier transform over the first register and the Hadamard transform on the second register and leave the third register unchanged. This gives the state
$$\frac{1}{\sqrt{2|A|}} \sum_{x \in A} |x\rangle|0\rangle|0\rangle + |x\rangle|1\rangle|0\rangle.$$
- Conditioning on the state in the second register, query functions f and g in the third register if the second register has state $|0\rangle$ and $|1\rangle$ respectively. The following state is obtained.
$$\frac{1}{\sqrt{2|A|}} \sum_{x \in A} |x\rangle|0\rangle|f(x)\rangle + |x\rangle|1\rangle|g(x)\rangle.$$
- Measure the last register to get the following state for some uniformly random $x \in A$
$$\frac{1}{\sqrt{2}}(|x\rangle|0\rangle + |x + s\rangle|1\rangle).$$
- Apply the quantum Fourier transform on the group $A \times \mathbb{Z}_2$ to obtain the state
$$\frac{1}{\sqrt{2|A|}} \sum_{\chi \in A^*} (\chi(x) + \chi(x+s))|\chi(x)\rangle|0\rangle + (\chi(x) - \chi(x+s))|\chi(x)\rangle|1\rangle$$
where χ is a character of A.

- Measure the state above. We are interested in getting only the outcome 1 in the second register and it is not hard to show its occurrence probability is $1/2$. Conditioning on the outcome of the second register to be 1, by repeating the process above, we see that the outcomes of measurement of the first register obey the promise on samples in RANDOM LINEAR INEQUATIONS(A, c) - search version.

It is then shown in [Ivanyos, 2008] that there is a reduction from the search version of RANDOM LINEAR INEQUATIONS(A, c) to its decision version defined below when the underlying group $A = \mathbb{Z}_p^n$. The set of characters of an abelian grou G, χ_u forms a multiplicative abelian group called character group since $(\chi_u \chi_{u'})(x) = \chi_u(x)\chi_{u'}(x)$ for all $x \in G$. In the following definition, A'^* is the character group of A' and A' is a subgroup of A.

Definition (RANDOM LINEAR INEQUATIONS(A', c') - decision version). Given samples of elements of A'^* from either

- a distribution that is c'-nearly uniform on characters not containing a fixed element s in their kernels OR
- a distribution that is nearly uniform on the whole A.

The goal is to decide which is the case.

For the special case when $A = \mathbb{Z}_m^n$, the reduction from the search version of RANDOM LINEAR INEQUATIONS(A, c) to its decision version can be seen in the following way. First, note that we are basically looking for some $s \in \mathbb{Z}_m^n$ which is the answer to the instance of the search version of RANDOM LINEAR INEQUATION. The idea is to guess the first component of s and observe if the new resulting set of equations has a solution or not. This confirms whether we have guessed correctly or not. We keep guessing the first component until we find the correct value for it. Then we fix this correct value in the equations and start guessing the second component. We do this recursively until all the components of s are revealed. It is noteworthy to say that since for each component, there are only constant number of possibilities, we can afford to check all our guesses.

The algorithm for deciding whether a set of equations has a solution or not when m is a prime number is basically the same as the one in [Friedl et al., 2003]. However, in [Friedl et al., 2003] we are finding such a solution, but here we are deciding if a set of equations has a solution or not. So the only thing we need to do is to compute the rank of equations left after doing linearization. For the case that m is not a prime number but a prime power, we use a mapping from the elements of $\mathbb{Z}_{p^k}^n$ to the elements of $\mathbb{Z}_p^{kn}$. The mapping is simply the linear p-base expansion. We show these new elements with an asterisk. Note that, for the old samples before doing linearization, we had

$$u \cdot s \neq 0.$$

According to the mapping above, for the new samples we cannot directly write

$$u^* \cdot s^* \neq 0$$

since we are ignoring the carries introduced. It can be shown that a corrective polynomial term $c(s_1^*, s_2^*, \ldots, s_{kn}^*)$ can be added to $u^* \cdot s^*$ to take the carry into account. Denote $P(s_1^*, s_2^*, \ldots, s_{kn}^*) = c(s_1^*, s_2^*, \ldots, s_{kn}^*) + u^* \cdot s^*$. Now we have a low degree polynomial inequation such that

$$P(s_1^*, s_2^*, \ldots, s_{kn}^*) \neq 0.$$

The polynomial is low degree since the corrective term added for the carry has a constant degree. Now one can use the Fermat's little theorem again to obtain a polynomial equation

$$(P(s_1^*, s_2^*, \ldots, s_{kn}^*))^{p-1} = 1.$$

Again, the idea is to linearize this new polynomial in terms of its monomials and decide if the rank of equations is complete or not by looking at a sufficient number of equations of the form above. In the work of Ivanyos [2008], having a constant c is crucial for having an efficient algorithm.

Chapter 3

Injectivization

In this chapter, we introduce a general purpose tool called injectivization for reducing the average case non-injective hidden shift problem to an instance of the injective hidden shift problem over the same group. More specifically, this process gets as input a non-injective function defined over an arbitrary group and with high probability over the choice of the function, outputs an injective function defined over the same group. For this tool to be useful for solving the non-injective hidden shift problem–by reduction to the injective hidden shift problem–we require that if two non-injective functions have some relative shift, this shift be preserved in their corresponding injectivized functions which are output by the injectivization process. Due to this constraint, simply concatenating values of f at arbitrary places–which might yield an injective function–might not be useful since the relative shift of the two functions injectivized in this way is not preserved. Hence, the main concern is finding a process under which, the algebraic shift structure is preserved between the two functions. The process described below offers a framework under which both goals of injectivizing functions and preserving the relative shift are met.

Our result is quite strong since we do not use any special structure of the different groups. In other words, functions can be defined over any groups. This is usually a limiting factor of the quantum algorithms that their use is only limited to the abelian case or the groups with certain structure. However, we do not require the groups to be abelian and also do not expect any special structure in them.

Another strength of the result is that we do not put any limitation on the range of the functions. Again this is a limiting factor for some quantum algorithm since they are sensitive to the range set of the oracles and they expect them to fulfil some requirements like being binary or etc. However, again, it is not the case here and our functions can have range in arbitrary sets.

Injectivization process:

> **Input**: any function $f : G \to S$ and an m-tuple $V \in G^m$.
>
> **Output**: function $\mathbf{f}_V : G \to S^m$ constructed in the following way:
>
> $$\mathbf{f}_V(x) := (f(x+v_1), f(x+v_2), \ldots, f(x+v_m))$$
>
> We denote the output of the injectivization process with boldface letters. We say that the injectivization succeeds if $\mathbf{f}_V$ is injective; otherwise it fails.

To show that injectivization fails only with small probability, in lemma 3.1 we show that the probability of a collision (i.e., $\mathbf{f}_V(x) = \mathbf{f}_V(y)$ for any fixed $x, y \in G$) for a uniformly random function f is small if V has distinct components. However, the proof is a bit complicated since random variables $\mathbf{f}_V(x)$ and $\mathbf{f}_V(y)$ are not necessarily independent which can be shown to result in a higher collision probability. Our original goal is to –by concatenating polynomially many values that f takes at various places– make the probability of a collision exponentially small. In other words, as stated in Corollary 3.4, a polynomially large m is enough for achieving this goal. It is implicitly shown in lemma 3.1 that to fix the loss from these interdependencies we need to concatenate twice as much evaluations of f at distinct places as it was needed for the case that $\mathbf{f}_V(x)$ and $\mathbf{f}_V(y)$ were truly independent.

Lemma 3.1. *For an arbitrary $V \in G^m$ with distinct components and for a uniformly random function $f : G \to S$, the probability that $\mathbf{f}_V$ is not injective is at most* $\dfrac{|G|^2}{|S|^{\lceil m/2 \rceil}}$.

Proof. We will show that, for any distinct x and y in the domain, $\Pr[\mathbf{f}_V(x) = \mathbf{f}_V(y)] \leq 1/|S|^{\lceil m/2 \rceil}$ and then the result follows from the union bound.

Let x and y be any two points in the domain of the function. If all of the components of $(x+v_1, x+v_2, \ldots, x+v_m)$ and $(y+v_1, y+v_2, \ldots, y+v_m)$ are distinct then it is clear that equality in each component is independent, so $\Pr[\mathbf{f}_V(x) = \mathbf{f}_V(y)] = 1/|S|^m$. However, the components need not all be distinct in which case there can be dependencies among components. To illustrate, consider the case where $G = \mathbb{Z}_2^n$, $|S| = 2$ and $m = 2$. If $x = v_1$ and $y = v_2$ then $(x+v_1, x+v_2) = (0, v_1 \oplus v_2)$ and $(y+v_1, y+v_2) = (v_1 \oplus v_2, 0)$, so a collision in the first component implies a collision in the second component. Therefore, the probability of the collision $\mathbf{f}_V(x) = \mathbf{f}_V(y)$ is $1/2$ rather than $1/4$.

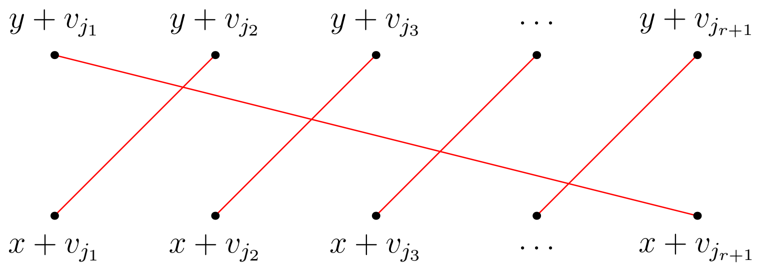

Figure 3.1: Components are shown with vertices and equal components are connected with an edge.

To address the general case, consider a maximal chain of dependencies (figure 3.1):

$$\begin{aligned} x + v_{j_1} &= y + v_{j_2} \\ x + v_{j_2} &= y + v_{j_3} \\ \vdots \quad & \quad \vdots \\ x + v_{j_r} &= y + v_{j_{r+1}}. \end{aligned}$$

This implies:

$$\begin{aligned} f(x + v_{j_1}) &= f(y + v_{j_2}) \\ f(x + v_{j_2}) &= f(y + v_{j_3}) \\ \vdots \quad & \quad \vdots \\ f(x + v_{j_r}) &= f(y + v_{j_{r+1}}). \end{aligned}$$

If $j_{r+1} = j_1$ then we have an r-cycle (the above example is a 2-cycle). Collisions in components $j_2, \ldots, j_r$ of $\mathbf{f}_V(x)$ and $\mathbf{f}_V(y)$ occur independently; however if all these components collide, a collision in the component j_1 is implied (figure 3.2).

Therefore, the probability of a collision among components $j_1, \ldots, j_r$ is $1/|S|^{r-1}$. If, on the other hand, the chain is not cyclic then the probability of a collision among components $j_1, \ldots, j_{r+1}$ is $1/|S|^{r+1}$. Since all maximal chains of dependencies are disjoint, the probability of $\mathbf{f}_V(x) = \mathbf{f}_V(y)$ is the highest when there are $m/2$ 2-cycles, when it is $1/|S|^{\lceil m/2 \rceil}$ (figure 3.3). It is so because when m is an odd number, the worst case happens when we have $(m-1)/2$ 2-cycles and the last components be independent and involved in no chains which gives a collision probability of $1/|S|^{(m-1)/2} \times 1/|S| = 1/|S|^{\lceil m/2 \rceil}$ as stated. $\square$

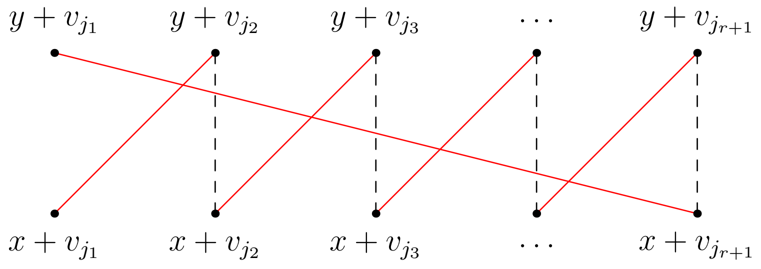

Figure 3.2: Components are shown with vertices and equal components are connected with an edge. Dashed lines show the accidental collisions.

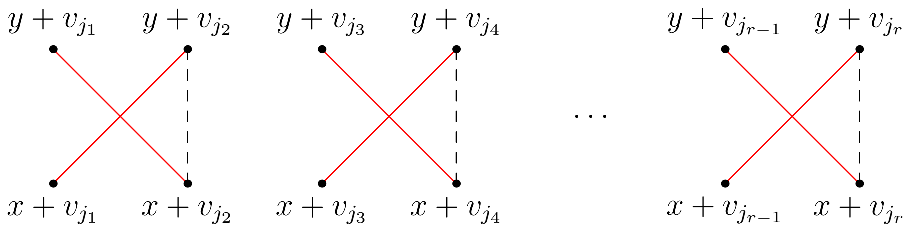

Figure 3.3: Components are shown with vertices and equal components are connected with an edge. Dashed lines show the accidental collisions.

We need the following simple lemma which demonstrates that our construction preserves the shift between the injectivized functions.

Lemma 3.2. *Given functions $f, g : G \to S$, for arbitrary V and all $x \in G$, it holds that $f(x) = g(x+s)$ for some $s \in G$ if and only if $\mathbf{f}_V(x) = \mathbf{g}_V(x+s)$.*

Proof. Intuitively the lemma must hold since $\mathbf{f}_V$ just consists of shifts of f and $\mathbf{g}_V$ has the corresponding shifts of g.

More formally, it is easy to see that if $f(x) = g(x+s)$ for all x and some s in G, then $\mathbf{f}_V(x) = \mathbf{g}_V(x+s)$. To prove the other direction, let $x' = x + v_1$. We know that $\mathbf{f}_V(x) = \mathbf{g}_V(x+s)$ for all $x \in G$. Hence this also needs to hold for the first component of $\mathbf{f}_V(x)$ and $\mathbf{g}_V(x+s)$. Therefore $f(x+v_1) = g(x+v_1+s)$. Replacing $x+v_1$ with x', we have $f(x') = g(x'+s)$ for all $x' \in G$. $\square$

Now we have all the tools we need to show a reduction from the non-injective hidden shift problem to the injective hidden shift problem. The reduction is classical.

Theorem 3.3. *Injectivization, when it succeeds, reduces a uniformly and randomly chosen instance of the non-injective hidden shift problem $f, g : G \to S$ to an instance of the hidden shift problem with functions $\mathbf{f}_V, \mathbf{g}_V : G \to S^m$ where m is the number of V's components. Injectivization fails with probability at most $\frac{|G|^2}{|S|^{\lceil m/2 \rceil}}$ over the choice of f.*

Proof. For a classical computer, it is clear how to simulate a query to any of $\mathbf{f}_V$ or $\mathbf{g}_V$ by making m queries to f or g respectively. To simulate the query $\mathbf{f}_V(x)$, we query $f(x+v_1), f(x+v_2), \ldots, f(x+v_m)$ and form the m-tuple

$$\mathbf{f}_V(x) = (f(x+v_1), f(x+v_2), \ldots, f(x+v_m)) \,.$$

Queries to $\mathbf{g}_V$ can be simulated similarly.

For a quantum computer, to make a query to $\mathbf{f}_V$ at place x, we prepare the state

$$|x\rangle\, |x+v_1, \ldots, x+v_m\rangle$$

and make m queries to f to prepare the state

$$|x\rangle\, |x+v_1, \ldots, x+v_m\rangle \bigotimes |f(x+v_1), \ldots, f(x+v_m)\rangle \,.$$

Then, by efficiently uncomputing the second register, we obtain the state

$$|x\rangle\, |f(x+v_1), \ldots, f(x+v_m)\rangle\,.$$

Queries to $\mathbf{g}_V$ can be simulated similarly. This makes the number of queries m times more.

Based on the lemma 3.2 stating that the relative shift of f and g is the same as the relative shift of $\mathbf{f}_V$ and $\mathbf{g}_V$, this instance of the injective hidden shift problem has the same solution as the instance of the non-injective hidden shift problem. The failure rate of the injectivization is upper bounded by $\dfrac{|G|^2}{|S|^{\lceil m/2 \rceil}}$ when f is chosen uniformly at random as it is shown in lemma 3.1. However, here we have the promise on f and g to have a unique shift. This is equivalent to saying that f is not periodic. However, note that when f is periodic, the injectivization fails to output an injective function. This can be seen by applying lemma 3.2 to functions f, g where g is defined to be $g(x) = f(x+v)$ where $v \neq 0$ is the period of f. Hence, given the promise on f, the injectivization has even a lower chance of failure when it is fed a uniform instance of the non-injective hidden shift problem than the bound that is obtained in lemma 3.1. $\square$

Using theorem 3.3, the following corollary is trivial for polynomially large m:

Corollary 3.4. *Let $V \in G^m$ be composed of m distinct components. Having $m \geq (4+\epsilon)\log_{|S|}|G|$ for an arbitrary constant $\epsilon > 0$, an instance of the non-injective hidden shift problem with $f, g : G \to S$ is efficiently reduced to an instance of the hidden shift problem $\mathbf{f}_V, \mathbf{g}_V : G \to S^m$ with exponentially small failure probability over the uniform random choice of f.*

Theorem 2 in [Gavinsky et al., 2011] states that by the algorithms in that paper, for solving an average case instance of the Boolean hidden shift problem, an exponential separation can be achieved. Here, we simplify this result in the theorem 3.5 by reducing the Boolean hidden shift problem to the Simon's problem. First, we state the Simon's problem here[Simon, 1994]:

Definition (Simon's problem). We are given oracle access to a function $h : \{0,1\}^n \to S$. Also, we have a promise on the function to have the following property: For any distinct x, y in the domain of the function

$$h(x) = h(y) \iff x \oplus y = s$$

where $s \in \{0,1\}^n$ is some fixed non-zero binary string. The goal of the problem is to find s.

Simon's problem is one of the very first problems discovered for which an efficient quantum algorithm exists while no efficient classical algorithm can solve the problem.

Theorem 3.5. *There is an efficient reduction from the Boolean hidden shift problem for random functions to the Simon's problem.*

Proof. We use injectivization over $f, g : \mathbb{Z}_2^n \to \{0, 1\}$ and then construct the oracle of the Simon's problem $h : \mathbb{Z}_2^{n+1} \to \{0, 1\}^m$ where $h(0x_n x_{n-1} \ldots x_1) = \mathbf{f}_V(x_n x_{n-1} \ldots x_1)$ and $h(1x_n x_{n-1} \ldots x_1) = \mathbf{g}_V(x_n x_{n-1} \ldots x_1)$.

Injectivization succeeds in outputting an injective function in the average case. However, in the Boolean hidden shift problem, functions are not average case since there is a promise on them that they do not have a self-shift. But this does not have any effect on our argument since injectivization always fails to output an injective function if the input function has a self-shift as described in more detail in the proof of theorem 3.3. Hence, by excluding those functions, injectivization even succeeds with a higher chance of success. $\square$

Gavinsky et al. [2011] posed an open question whether the methods they have used for solving the hidden shift over $\mathbb{Z}_2^n$ for the Boolean functions can be generalized and adapted for the case of the non-Boolean functions. This question may be interpreted in different ways. We have not used the method in [Gavinsky et al., 2011]. But we answer the question using our own method and having the following interpretations. The most basic generalization is to keep the group unchanged but let the functions in the hidden shift problem take also values other than $\{0, 1\}$. We already showed in our theorem 3.5 how to reduce such a problem to the Simon's problem. The most general format of the open question we could imagine is to consider the function over the other similar groups like the groups $\mathbb{Z}_q^n$ for $q \geq 3$. Fortunately, as noted in chapter 2, the injective hidden shift problem over those groups when q is a constant prime power is already addressed in [Friedl et al., 2003] and [Ivanyos, 2008]. Hence, we can use injectivization to reduce the problem to the already solved injective case.

Chapter 4

The hidden shift problem for c-almost generalized bent functions

In this chapter, we give a quantum algorithm which uses a radically different approach than injectivization and is based on the Fourier analysis. This algorithm can be understood as a generalization of the algorithm in [Gavinsky et al., 2011] for addressing the average case hidden shift problem. However, in this section, we do not address the average case hidden shift problem but rather the problem when the underlying function is from the class of c-almost generalized bent functions (which includes the generalized bent functions) instead of a uniformly random function.

The hidden shift problem for bent functions is already addressed in the [Gavinsky et al., 2011]. The first partial progress with the hidden shift problem for bent functions appeared in [Rötteler, 2010] where some classes of bent functions were addressed.

Note that it is presumably not easy to say if a generalized bent function or more generally a c-almost generalized bent function can be injectivized. Hence, it is not clear whether the injectivization method can be used for solving the hidden shift problem for these functions. In general, it seems to be a very hard problem to fully characterize the set of functions which can be injectivized unless there exists some uniformity condition on the functions. One example of such condition is the function to be drawn uniformly at random.

Since in most applications, we are concerned about generalized bent functions over $\mathbb{Z}_p^n$ only when p is a prime number, we illustrate the algorithm solving generalized bent functions only in that case. Our algorithm is efficient only if the prime number is constant.

Let $f, g : \mathbb{Z}_p^n \to \mathbb{Z}_p$ define an instance of the non-injective hidden shift problem and be generalized bent functions or any functions from the superset c-almost generalized bent functions. Then the following sampling subroutine can be used to efficiently obtain polynomially many random linear inequations. Without loss of generality, we assume $s \neq 0$. We can easily check if $s = 0$ in which case we are done if it happens that $s = 0$.

Quantum sampling(f, g)

Step 1: Prepare the initial state in $|0\rangle|0\rangle$. Apply the quantum Fourier transform over $\mathbb{Z}_p^n \times \mathbb{Z}_2$ to obtain the state

$$\frac{1}{\sqrt{2N}} \sum_{x \in \mathbb{Z}_p^n} (|x\rangle|0\rangle + |x\rangle|1\rangle) .$$

Step 2: Using this state, query (in phase) oracles f if the state of the second register is $|0\rangle$ and g if it is $|1\rangle$. This transforms the state into

$$\frac{1}{\sqrt{2N}} \sum_{x \in \mathbb{Z}_p^n} (F(x)\,|x\rangle|0\rangle + G(x)\,|x\rangle|1\rangle) =$$

$$\frac{1}{\sqrt{2N}} \sum_{x \in \mathbb{Z}_p^n} (F(x)\,|x\rangle|0\rangle + F(x-s)\,|x\rangle|1\rangle) .$$

Step 3: Apply quantum Fourier transform over $\mathbb{Z}_p^n \times \mathbb{Z}_2$ again.

$$\frac{1}{\sqrt{2}} \sum_{u \in \mathbb{Z}_p^n} \left(\hat{F}\,(u) \left(1 + \omega_p^{-u \cdot s}\right) |u\rangle|0\rangle + \hat{F}\,(u) \left(1 - \omega_p^{-u \cdot s}\right) |u\rangle|1\rangle \right) .$$

Step 4: Measure the state. Output nothing if the outcome in the second register is 0. Otherwise output the outcome of the first register. This will always output some $u \in \mathbb{Z}_p^n$ such that

$$u \cdot s \neq 0.$$

It is not hard to prove that the quantum sampling subroutine outputs 1 as the outcome of its second register with constant probability:

Theorem 4.1. *Quantum sampling(f, g) outputs 1 with constant probability in its second register upon measurement, if function $f : \mathbb{Z}_p^n$ is a c-almost generalized bent function.*

Proof. First consider the case of generalized bent functions. The probability of the outcome 1 in the second register is

$$\frac{1}{N}\sum_{u\in\mathbb{Z}_p^n}\frac{\left|1-\omega_p^{u\cdot s}\right|^2}{\left|1+\omega_p^{u\cdot s}\right|^2+\left|1-\omega_p^{u\cdot s}\right|^2}=\frac{1}{N}\sum_{u\in\mathbb{Z}_p^n}\frac{2-\omega_p^{u\cdot s}-\left(\omega_p^{u\cdot s}\right)^*}{4}=\frac{1}{2}.$$

Hence, for generalized bent functions, the desired outcome happens with probability 1/2 which is a constant. This result can be extended to the case of c-almost generalized bent functions in the following way: Since we hope to get outcome 1 in the second register, the worst case happens when the following condition is met: If for any u the absolute value of the amplitude of $|u,1\rangle$ is greater than that of $|u,0\rangle$, then $|\hat{F}(u)|$ be the minimum and otherwise $|\hat{F}(u)|$ be the maximum.

However, this only changes the probability of getting outcome 1 by a constant factor due to the condition that we have that

$$\frac{\min_x\left|\hat{F}(x)\right|}{\max_y\left|\hat{F}(y)\right|}=c.$$

□

Now similar to [Friedl et al., 2003], we can use the idea of linearization to find the hidden shift. To do so, we first raise both sides of each inequality obtained in step 4, to the power $p-1$. This gives a set of polynomial equations of form

$$\begin{aligned}(u_1\cdot s)^{p-1} &= 1\\ (u_2\cdot s)^{p-1} &= 1\\ \vdots\quad &\quad \vdots\\ (u_k\cdot s)^{p-1} &= 1\end{aligned}$$

where the right side of each equality is 1 due to Fermat's little theorem. Then we use the well-known linearization technique which, in short, involves giving each monomial in the polynomial equations above a new variable name (and hence disregard the dependence between different monomials) and solving the resulting set of linear equations using the Gaussian elimination.

We show in the following theorem that for the case of generalized bent functions there exists an efficient process for getting a full rank set of linear equations and hence it is possible to find the unknown s using the Gaussian elimination algorithm.

Theorem 4.2. *When f is a generalized bent function, there is an efficient quantum algorithm for finding a full rank set of linear equations in s.*

Proof. In their sampling subroutine, the state before final observation in [Friedl et al., 2003] is

$$\frac{1}{\sqrt{2N}} \sum_{u \in \mathbb{Z}_p^n} \omega^{u \cdot x} \cdot (1 - \omega_p^{u \cdot s}) \left| u \right\rangle .$$

The state that we have before the final observation in our quantum sampling subroutine is

$$\frac{1}{\sqrt{2}} \sum_{u \in \mathbb{Z}_p^n} \hat{F}(u) \cdot (1 - \omega_p^{u \cdot s}) \left| u \right\rangle .$$

It is not hard to see that measuring in the computational basis produces the same statistics for both states if $\hat{F}(u)$ has a constant absolute value or in other words f is a generalized bent function. Hence, by the result in [Friedl et al., 2003], the rank of the linear equations will be complete after getting polynomially many samples since the probability of the outcomes upon measurement on our quantum state is equal to that of [Friedl et al., 2003]. Hence, this proves that, when our functions are generalized bent functions, we can efficiently find the hidden shift. □

We also show that, if f is a c-almost generalized bent function, then one can find the hidden shift by a probabilistic argument. In the following theorem, we show that, due to the nice properties of c-almost generalized bent functions, one can essentially use the same algorithm for c-almost generalized bent functions as the one for generalized bent functions and this will introduce only constant overhead.

Theorem 4.3. *When f is a c-almost generalized bent function, there is an efficient quantum algorithm for finding a full rank set of linear equations in s.*

Proof. To see this, consider the worst case where we already have $m-1$ linearly independent equations after doing linearization on the existing equations and we need only one more sample $u \in \mathbb{Z}_p^n$ from our sampling subroutine such that it makes the rank of the linear equations complete (after doing linearization on the resulting polynomial equation from that sample). First imagine that we did not have the factor $\hat{F}(u)$ (i.e. our quantum state before the final observation was exactly the one in [Friedl et al., 2003]). We are looking for some u from the set of all good outcomes $U \subset \mathbb{Z}_p^n$ which makes the rank of equations complete. This must happen with probability polynomially small in n in the worst case since it is proved in [Friedl et al., 2003] that their quantum algorithm is efficient. However, this image is not quite correct since the existence of $\hat{F}(u)$ changes the probability of getting an outcome from U. Hence we need to justify that in the general case things are not very different. To see this, note that the existence of $\hat{F}(u)$ changes the probability of a desired outcome only by a constant factor since the ratio between the smallest to biggest absolute value of $\hat{F}(u)$ over all u is some constant $c > 0$. Hence, if in the first case, one needs to get $p(n)$ samples to have a high chance of obtaining the desirable outcome (i.e. some u from U) , in the second case one will need to get $p(n)/c$ samples to have at least the same chance of obtaining an outcome from U. Since this is only a constant factor, the query and time complexity of the original algorithm will change only by a constant factor which is not important from a complexity theoretical perspective. $\square$

Chapter 5

Generalized hidden shift problem

The generalized hidden shift problem is defined in [Childs and van Dam, 2007] in the spirit of finding a problem which is easier than the hidden shift problem for quantum computers but still classically hard as it is shown in the next chapter. The generalized hidden shift problem is the task of finding the hidden shift $s \in \mathbb{Z}_N$ by extracting information from an oracle $f : \{0, 1, \ldots, M-1\} \times \mathbb{Z}_N \to S$ where S is a finite set when it holds that (a) for any fixed b, the function $f(b, x)$ is injective and (b) $f(b, x) = f(b+1, x+s)$ for all b and x in the domain. In [Childs and van Dam, 2007], this problem is originally defined over the group $\mathbb{Z}_N$ to interpolate between the dihedral HSP which is equivalent to the hidden shift problem over $\mathbb{Z}_N$ (i.e. when we are given only two oracles $M = 2$ [Ettinger and Høyer, 1999b][Kuperberg, 2005]) and abelian HSP. The main motive was to understand the difficulty of the dihedral HSP better.

When $M = N$, this problem is basically equivalent to the HSP over the abelian group $\mathbb{Z}_N \times \mathbb{Z}_N$ which we know how to solve efficiently by abelian Fourier sampling [Childs and van Dam, 2010]. When $M = 2$, the generalized hidden shift problem is the same as the hidden shift problem over $\mathbb{Z}_N$ which is equivalent to HSP over dihedral group. In [Childs and van Dam, 2007], it is shown that the generalized hidden shift problem can be solved efficiently provided $M = N^\epsilon$ where $0 < \epsilon < 1$ is a constant.

Not every instance of the generalized hidden shift problem defined in the sense above is hard for a classical computer. In particular it can be seen that for the case that N is a power of 2 (i.e. $N = 2^k$ for some k), a classical computer can efficiently solve this problem if it is given $M = N$ oracles. This can be seen by running the following classical algorithm (we use the binary expansion of b and s):

Step 1. Set $b_n b_{n-1} \ldots b_1 \leftarrow 10\ldots0$ and $s_n s_{n-1} \ldots s_1 \leftarrow 00\ldots0$ and $i \leftarrow n$

Step 2. Make a classical query to find

$$q_1 \leftarrow f(00\ldots0, 00\ldots0)$$

Step 3. Make a classical query to find

$$q_2 \leftarrow f(b_n b_{n-1}\ldots b_1, s_n s_{n-1}\ldots s_1)$$

Step 4. If $q_1 = q_2$ then $s_{(n+1)-i} \leftarrow 0$ otherwise $s_{(n+1)-i} \leftarrow 1$.

Step 5. if $i = 1$ then output $s_n s_{n-1}\ldots s_1$ and we are done, otherwise $i \leftarrow i-1$

Step 6. Set $b_j \leftarrow 0$ for all $j \neq i$ and $b_i \leftarrow 1$. Go to step 3.

An algorithm with the same essence is pointed out in [Niel de Beaudrap et al., 2002].

Similarly, we define a generalized hidden shift problem over groups of form $\mathbb{Z}_q^n$ especially since no method is known to solve the hidden shift problem when q is a composite number that is not a prime power. Even in the smallest case, $q = 6$, the problem is still open.

It is interesting to point out that unlike the original generalized hidden shift problem that is easy in some cases such as when $N = 2^n$ (as it can be seen by the efficiency of the algorithm above), as we show in chapter 6, the generalized hidden shift problem for groups of form $\mathbb{Z}_q^n$ when q is a constant number is always classically hard.

So far, we know how to solve the hidden shift problem when q is a constant prime power based on the works of [Friedl et al., 2003] and [Ivanyos, 2008]. We can use the following method to solve the generalized hidden shift problem over the group $\mathbb{Z}_q^n$ for any constant composite number q. Given $q = p_1^{k_1} \times p_2^{k_2} \times \cdots \times p_t^{k_t}$ is a factorization of q into its prime components $p_1 < p_2 < \cdots < p_t$, we give a quantum algorithm that only needs oracles $f(b, x)$ for which $b \in \{0\} \cup \{q/p_1^{k_1}, q/p_2^{k_2}, \ldots, q/p_t^{k_t}\}$. Calling each of the oracles obtained by fixing b, a shifted oracle, $t+1$ shifted oracles are used due to this algorithm where t is the number of distinct prime divisors of q. It is interesting as an open question to know whether one can solve this problem using fewer oracles. We use these oracles in the following way in the following theorem.

Theorem 5.1. *There is an efficient quantum algorithm which solves the generalized hidden shift problem with an underlying group $\mathbb{Z}_q^n$ where q is a constant number using only the oracles obtained by the fixed values of $b \in \{0\} \cup \{q/p_1^{k_1}, q/p_2^{k_2}, \ldots, q/p_t^{k_t}\}$.*

Proof. The algorithm is as follows. Using algorithms in [Friedl et al., 2003] and [Ivanyos, 2008], we first solve the instance of the hidden shift problem defined by $g_1, g_2 : \mathbb{Z}^n_{p_1^{k_1}} \to S$ where $g_1(x) = f(0, y(x))$ and $y : \mathbb{Z}^n_{p_1^{k_1}} \to \mathbb{Z}^n_q$ maps x to the unique element of $\mathbb{Z}^n_q$ for which each component has remainder 0 to $p_j^{k_j}$ for $2 \le j \le t$ and component y_i has remainder x_i to $p_1^{k_1}$. In other words, $y(x)$ is (by Chinese remainder theorem) the unique element which satisfies

$$\begin{cases} y(x) = x & \mod p_1^{k_1} \\ y(x) = 0 & \mod p_2^{k_2} \\ \quad\vdots & \\ y(x) = 0 & \mod p_t^{k_t} \end{cases}$$

where $\mod$ is understood component-wise.

We define g_2 in a similar way, only replacing $f(0, y)$ with $f(q/p_1^{k_1}, y)$ in its definition: $g_2(x) = f(q/p_1^{k_1}, y(x))$. It is not hard to see that $g_1(x) = g_2(x + s')$ for all x in the domain where for $s' \in \mathbb{Z}^n_{p_1^{k_1}}$ we have

$$s' = s \mod p_1^{k_1}.$$

By solving the hidden shift problem defined similarly using oracles $f(0, y)$ and $f(q/p_i^{k_i}, y)$ for all $1 \le i \le t$, one can find $(s \mod p_2^{k_2}), \ldots, (s \mod p_t^{k_t})$. Hence we have the following set of equations

$$\begin{aligned} s &= c_1 \mod p_1^{k_1} \\ s &= c_2 \mod p_2^{k_2} \\ \vdots &\quad \vdots \\ s &= c_t \mod p_t^{k_t}. \end{aligned}$$

Since $p_1^{k_1}, p_2^{k_2}, \ldots, p_t^{k_t}$ are all mutually coprime, for each component, by Chinese remainder theorem, we are guaranteed that there exists a unique solution to this set of equations and indeed an efficient algorithm exists that gives that solution. $\square$

As shown above, for a constant composite number we only need as many extra shifted oracles (other than $f(0, x)$) as the number of distinct prime divisors of q. As a toy example, for the case that $q = 6$, we will only need the shifted oracles for which $b = 0$, $b = 2$ and $b = 3$. We can use the functions with $b = 0$ and $b = 2$ to find $(s \mod 3)$. Then to uniquely determine s, we can use the functions with $b = 0$ and $b = 3$ to find $s \mod 2$. Applying Chinese remainder theorem for each component of s, guarantees finding s mod 6 uniquely.

As another example, for finding s for the case that $q = 12$, we use the oracles for which $b = 0$, $b = 3$ and $b = 4$. Then, we solve the hidden shift problem defined in the above sense using functions with fixed $b = 0$ and $b = 3$ to find $(s \mod 4)$. We also use the functions with $b = 0$ and $b = 4$ to find $(s \mod 3)$. By applying Chinese remainder theorem to each components of s, we can find the unique solution to s.

The above procedure gives improvement over the following simpler procedure which requires all the possible shifted oracles to be provided:

Quantum sampling

Step 1: Prepare the initial state in $|0, 0\rangle|0\rangle$.

Step 2: Apply the quantum Fourier transform over $\mathbb{Z}_q \times \mathbb{Z}_q^n$ on the first register. This gives the state

$$\frac{1}{\sqrt{q^{n+1}}} \sum_{x \in \mathbb{Z}_q^n} \left(|0, x\rangle |0\rangle + |1, x\rangle |0\rangle + \cdots + |q-1, x\rangle |0\rangle\right).$$

Step 3: Query the oracle f. Denote $f'(x) = f(0, x)$. This transforms the state into

$$\frac{1}{\sqrt{q^{n+1}}} \sum_{x \in \mathbb{Z}_q^n} |0, x\rangle |f'(x)\rangle + |1, x\rangle |f'(x+s)\rangle + \cdots + |q-1, x\rangle |f'(x + (q-1)s)\rangle.$$

Step 4: Measure the last register. For some random x, this transforms the state into

$$\frac{1}{\sqrt{q}}(|0, x\rangle + |1, x+s\rangle + \cdots + |q-1, x+(q-1)s\rangle).$$

Step 5: Apply the quantum Fourier transform over $\mathbb{Z}_q \times \mathbb{Z}_q^n$. This gives the state

$$\frac{1}{\sqrt{q^{n+2}}} \sum_{u \in \mathbb{Z}_q^n, k \in \mathbb{Z}_q} \omega_q^{u \cdot x} \left(1 + \omega_q^{k + u \cdot s} + \cdots + \omega_q^{(q-1)(k+u\cdot s)}\right) |k, u\rangle$$

Step 6: Measure the state. This always gives some u and k for which it holds that

$$u \cdot s = -k$$

Being given enough equations from step 6 above, one can solve the set of equations. Note that the Gaussian elimination algorithm cannot be used since $\mathbb{Z}_q$ is not a field. Instead, we can convert the matrix of coefficients in the equations to the Smith normal form since $\mathbb{Z}_q$ is a principal ideal domain. The resulting diagonal matrix is surely invertible (each of its element on the diagonal is invertible) since we have a unique solution. To show that the above procedure is efficient, we need to show that the set of samples u drawn in step 6 span the whole space with high probability after polynomially many iterations.

Theorem 5.2. *The set of $u_1, u_2, \ldots, u_t \in \mathbb{Z}_q^n$ drawn from the quantum sampling subroutine above, span $\mathbb{Z}_q^n$ with high probability if $t \geq n^2$.*

Proof. First observe that the state of the system after the step 5 (i.e. the state before the final observation) is:

$$\frac{1}{\sqrt{q^{n+2}}} \sum_{u \in \mathbb{Z}_q^n, k \in \mathbb{Z}_q} \omega_q^{u \cdot x} \left(1 + \omega_q^{k+u \cdot s} + \cdots + \omega_q^{(q-1)(k+u \cdot s)}\right) |k, u\rangle =$$
$$\frac{1}{\sqrt{q^n}} \sum_{u \in \mathbb{Z}_q^n} \omega_q^{u \cdot x} \left|-u \cdot s, u\right\rangle .$$

It is clear that all $u \in \mathbb{Z}_q^n$ are equally likely to occur. The worst case is when we already have $n-1$ linearly independent vectors and we are trying to find the last vector with the property that it be linearly independent from the previous vectors. Assuming v is the vector orthogonal to the space spanned by the $n-1$ linearly independent vectors $u_1, u_2, \ldots, u_{n-1}$ that we already have; we want our last vector u_n to have non-zero inner product with v. We claim that this happens with probability at least $1/q$: v has at least one non-zero component which we denote with v'. We denote with v'' the vector that is obtained by omitting this component. The inner product between a vector u that is drawn uniformly at random and v is obtained by the following:

$$u \cdot v = u' \cdot v' + u''v''$$

For any fixed u' and v', as u'' ranges over $\mathbb{Z}_q$, $u \cdot v$ takes at least two distinct values (obviously at least one of them is non-zero). This is due to the facts that firstly, the first term above (i.e. $u' \cdot v'$) is independent from u'' and v'' and secondly, $u''v'' = 0$ if $u'' = 0$ and $u''v'' = v''$ if $u'' = 1$.

Given n random samples drawn from the quantum sampling subroutine for the purpose of finding the last linearly independent vector, the probability of failure in finding a vector with non-zero inner product with v is at most $(1 - 1/q)^n$.

Now, looking at the whole process of finding n linearly independent vectors, finding the first linearly independent vector is trivial. Any non-zero vector from the quantum sampling subroutine will fit. Using only one sample, the probability of a failure is $1/q^n$. For finding the next $n-1$ linearly independent vectors, we know from the calculations above and union bound that the probability of a failure is at most $(n-1)(1-1/q)^n + 1/q^n$ which is exponentially small in n. □

In the original generalized hidden shift problem in [Childs and van Dam, 2007], as mentioned above, when $M = N$, we have an instance of the abelian hidden subgroup problem which we know how to solve efficiently. It can be shown that a similar result holds for the generalized hidden shift problem over group $\mathbb{Z}_q^n$. When we are given all the q shifted oracles, we are basically given an instance of the hidden subgroup problem over group $\mathbb{Z}_q^{n+1}$. To show this, first, we formally define the abelian hidden subgroup problem and then show in theorem 5.3 that the generalized hidden shift problem over $\mathbb{Z}_q^n$ is an instance of this problem.

The abelian hidden subgroup problem can be solved efficiently using abelian Fourier sampling [Childs and van Dam, 2010]. The above algorithm which is presented in a simpler form, follows the same steps as this standard procedure for solving the generalized hidden shift problem with all possible shifted oracles. In the next chapter we prove that even this problem is still hard for a classical computer.

Definition (Abelian hidden subgroup problem). Given a black box for function $f : A \to S$ where A is any abelian group additively denoted and S is an arbitrary finite set, and given the promise that the following holds for the function f and some subgroup $H \leq A$, find a generating set for the subgroup H by making queries to f:

$$f(x) = f(y) \Longleftrightarrow x - y \in H.$$

It is not hard to see that the following result holds:

Theorem 5.3. *The generalized hidden shift problem over $\mathbb{Z}_q^n$ with $M = q$ is an instance of the abelian hidden subgroup problem over $\mathbb{Z}_q \times \mathbb{Z}_q^n = \mathbb{Z}_q^{n+1}$.*

Proof. The function $f : \mathbb{Z}_q \times \mathbb{Z}_q^n \to S$ defined in the generalized hidden shift problem is injective when its first argument $b \in \mathbb{Z}_q$ is fixed. It is easy to see that if $x - y \in H$ where H is the subgroup generated by

$$(1, s) \in \mathbb{Z}_q \times \mathbb{Z}_q^n = \mathbb{Z}_q^{n+1}$$

and s is the solution to the instance of the generalized hidden shift problem, then $f(x) = f(y)$. In the other direction, we need to show that if for any distinct x, y it holds that $f(x) = f(y)$ then $x - y \in H$ or equivalently if $x - y \notin H$, then $f(x) \neq f(y)$. Assume that $x = (k_1, k_1 s + r_1)$ and $y = (k_2, k_2 s + r_2)$. Hence, $f(0, r_1) = f(x)$ and $f(0, r_2) = f(y)$. Note that, r_1 cannot equal r_2, otherwise the assumption $x - y \notin H$ is violated. Since for any fixed b, the function $f(b, t)$ on t is injective, $f(x) \neq f(y)$. $\square$

Chapter 6

Classical complexity

In this chapter, we show that, the classical query complexity of the non-injective hidden shift problem when the underlying group is $\mathbb{Z}_m^n$ is high in the average case. For proving this bound, we are benefited from some of the ideas in [Gavinsky et al., 2011].

Also, we prove the injective hidden shift problem and its generalized counterpart are hard classically. Interestingly, unlike the original generalized hidden shift problem with the underlying group $\mathbb{Z}_N$ which is not hard for certain values of N as shown in the previous chapter (although the corresponding hidden shift problem is still hard), the generalized hidden shift problem over the group $\mathbb{Z}_q^n$ with constant q is always hard classically.

First, we define an artificial version of the non-injective hidden shift problem. This problem will help us in proving the classical lower bounds for non-injective hidden shift problem.

Definition (No-promise non-injective random Hidden Shift Problem). Pick $s \in G$ and oracle $f : G \to S$ uniformly at random. Construct oracle $g : G \to S$ according to $g(x) = f(x - s)$ for all $x \in G$. The goal of the problem is to find the *hidden shift* s given oracles f and g.

It can be easily seen from the definition above that when f and g happen to be periodic functions, information theoretically it is not possible to uniquely output s. Hence, all the algorithms, no matter how many queries they make are in disadvantage for finding s in those cases and all shifts consistent with the structure of f and g are equally likely to be the correct one.

Theorem 6.1. *In order to achieve a success probability of at least* $1/2$ *in finding the solution to the no-promise non-injective hidden shift problem defined with a solution* $s \in \mathbb{Z}_q^n$

and functions $f, g : \mathbb{Z}_q^n \to S$, at least $\Omega\left(p_1^{n/2}\right)$ queries are needed when q is a constant number and p_1 is the smallest prime divisor of q.

Proof. First we assume that q is a prime number, then we address the general case. Without loss of generality, in this case, we assume s to be non-zero. We assume that queries are made in the form $(f(x), g(x))$.

We assume that the classical computer is given a magical bell which works as follows: whenever queries are made in the places X_1 and X_2, the magical bell rings if for any non-zero $d \in \mathbb{Z}_q$ it holds that $s = d(X_2 - X_1)$.

Let $S'_1, S'_2, \ldots, S'_m$ be an enumeration of all 1-dimensional subspaces of $\mathbb{Z}_q^n$. It is not hard to verify that the number of such sets is $m = \dfrac{q^n - 1}{q - 1}$: Since q is a prime number and so every non-zero element in $\mathbb{Z}_q$ has an inverse, if any two 1-dimensional subspaces have one non-zero common element, then they are identical. Each of the sets above have q elements and the only common element between each two sets is zero. Since each non-zero element in $\mathbb{Z}_q^n$ is contained in only one of the sets above and there are $q^n - 1$ such elements, it follows that there are $\dfrac{q^n - 1}{q - 1}$ such sets. Given the argument above, it is not hard to see that the sets $S_1, S_2, \ldots, S_m$ obtained by removing 0 from the sets above are disjoint and each have $q - 1$ elements. In other words, each S_i is of the form

$$\{d \cdot a | d \in \mathbb{Z}_q \setminus \{0\}\}$$

for some non-zero element $a \in \mathbb{Z}_q^n$.

We set our goal to be finding the set S_i corresponding to the hidden shift s. Since there is an obvious reduction from finding s to identifying the corresponding set, if the latter problem is hard, so is the former.

We explain the motivation behind working with the sets above. To do so, for a moment, assume there is no magical bell. The motivation is that in the case that $q = 2$, by querying at places X_1 and X_2, we reveal the values $f(X_1), f(X_2), g(X_1), g(X_2)$. If we see the pair $f(X_1)$ and $g(X_2)$ are equal, then it is possible that $s = X_2 - X_1$ be the hidden shift. However, we can also check if the pair $f(X_2)$ and $g(X_1)$ are equal in which case it is possible that $s = X_1 - X_2$ be the hidden shift. By making polynomially many more queries, we can verify with high probability if any of these two is the case. However, note that since $q = 2$, $X_2 - X_1 = X_1 - X_2$ holds. But this is not the case if we look at the case when $q = 3$. In that case, if we query at places X_1 and X_2, we are effectively guessing if $s = X_1 - X_2$ or $s = X_2 - X_1 = 2(X_1 - X_2)$. In other words, we are guessing if s is any non-zero multiple of $X_1 - X_2$. Hence, considering finding the set of non-zero multiples of s rather than s itself, gives us a handle to prove the result for all prime numbers q with one shot, instead of considering them on a case by case basis.

Since the sets $S_1, \dots, S_m$ are disjoint, an important observation is that knowing to which sets s does not belong, does not give any information about to which other set s might belong . Formally, let $Q_k := \{X_1, X_2, \dots, X_k\}$ be the places in which the queries are made after k queries. Also, let D be the set of all i's for which we know s does not belong to S_i according to our queries and the magical bell. Then

$$\Pr\left[s \in S_i | i \notin D\right] = \frac{1}{m - |D|} < \frac{1}{\dfrac{q^n - 1}{q - 1} - k^2}$$

Since conditioning on the queries does not provide any information, the best algorithm is to just random guess the set to which s belongs. As used in the above inequality, the best a classical computer is able to do is to eliminate $\binom{k}{2} < k^2$ possible sets after k queries. Hence, to be able to find the set which corresponds to s with probability at least 1/2, it needs to make at least $\Omega\left(q^{n/2}\right)$ queries.

Now, we turn to the case of composite numbers, where $q = p_1^{k_1} \times p_2^{k_2} \cdots \times p_t^{k_t}$ denotes the prime factorization of q where $p_1 < p_2 < \cdots < p_t$ holds.

Let the disjoint sets $T_1, T_2, \dots, T_l$ be an enumeration of the elements of $\mathbb{Z}_q^n$ which yield the same element if we apply the operator $(\mod p_1)$ to them component-wise. It is not hard to see that $l = p_1^n$. In other words, each T_i is of the form

$$T_i := \left\{x | x = a \mod p_1, x \in \mathbb{Z}_q^n\right\}.$$

for some $a \in \mathbb{Z}_{p_1}^n$. We define $S_1, S_2, \dots, S_m$ in an analogous way to their definition in the case of a prime q. We let them be the union of the sets T_i above in a way that each set S_j is consisting of only those sets T_i for which their corresponding element a_i are multiples of each other in $\mathbb{Z}_{p_1}^n$. In other words, each set S_j is of form

$$\left\{x | x = a \mod p_1, x \in \mathbb{Z}_q^n\right\} \cup \cdots \cup \left\{x | x = (p-1)a \mod p_1, x \in \mathbb{Z}_q^n\right\}$$

for some non-zero element $a \in \mathbb{Z}_{p_1}^n$. Again, it can be seen that $m = \dfrac{p_1^n - 1}{p_1 - 1}$. Without loss of generality, we assume

$$s \notin \left\{x | x = 0 \mod p_1, x \in \mathbb{Z}_q^n\right\}$$

since it happens with exponentially small probability.

Again, we have the magical bell which let the classical computer know if s belongs to S_i if it makes the queries X_1 and X_2 such that $X_1 - X_2 \in S_i$.

We update the definition of D with the new sets S_i and let it be the set of all i's for which we know s does not belong to S_i. As above, knowing to which sets s does not belong, does not give any extra information about the set to which s might belong. We formally have

$$\Pr\left[s \in S_i | i \notin D\right] = \frac{1}{m - |D|} < \frac{1}{\frac{p_1^n - 1}{p_1 - 1} - k^2}$$

Hence, to be able to find the set $S_i \ni s$ with probability at least $1/2$, one needs to make at least $\Omega\left(p_1^{n/2}\right)$ queries.

□

Theorem 6.2. *To solve the non-injective hidden shift problem for functions $f, g : Z_q^n \to S$ classically, $\Omega\left(p_1^{n/2}\right)$ queries are needed in the average case.*

Proof. The probability of f being periodic is very small. More formally, for any fixed $0 \neq r \in \mathbb{Z}_q^n$ and for all $x \in \mathbb{Z}_q^n$, it holds that

$$\Pr[f(x+r) = f(x)] \leq \frac{1}{2^{q^n/2}}$$

where the probability of the event is the highest when the order of r is 2. Hence, by the union bound, the probability of having a periodic function is at most $\frac{q^n}{2^{q^n/2}}$ which is double exponentially small and hence does not change the bound obtained above. Since the number of periodic functions is $o(1)$, the result of the theorem 6.1 also holds for the average case non-injective hidden shift problem for which we have the promise on the functions not to be periodic. □

So far in this chapter, we have dealt with the query complexity of the non-injective hidden shift problem. Now, we turn our attention toward the complexity of generalized injective hidden shift problem. We prove that this problem has a high classical query complexity and from that we deduce that the injective hidden shift problem must be hard too.

As it was proved in theorem 5.3, the generalized injective hidden shift problem is an instance of the abelian hidden subgroup problem with some promise on the hidden subgroup

$H \leq \mathbb{Z}_q \times \mathbb{Z}_q^n = \mathbb{Z}_q^{n+1}$ to be mentioned in theorem 6.4. We use the following theorem from the literature without mentioning its proof, which states the classical query complexity of the abelian hidden subgroup problem has a lower bound that depends on the number of only trivially intersecting subgroups [Childs and van Dam, 2010][Niel de Beaudrap et al., 2002].

Theorem 6.3. *If there exist k subgroups of G with only trivial pairwise intersections, then a classical computer has to make $\Omega(\sqrt{k})$ queries to solve the hidden subgroup problem when the underlying group is G.*

Now we show that for the generalized hidden shift problem expressed in terms of the abelian hidden subgroup problem, there are exponentially many non-trivially intersecting subgroups and hence by the previous theorem, its classical query complexity is exponential.

Theorem 6.4. *Let $f : \mathbb{Z}_q \times \mathbb{Z}_q^n \to S$ define an instance of the generalized hidden shift problem over $\mathbb{Z}_q^n$ where q is a constant number. Then the classical query complexity of the generalized hidden shift problem is exponential in $\log |G|$ where $G = \mathbb{Z}_q^n$.*

Proof. First, observe that by the theorem 5.3, the generalized hidden shift problem is equivalent to the hidden subgroup problem when the underlying group is $\mathbb{Z}_q \times \mathbb{Z}_q^n$ and the hidden subgroup is of form

$$H = \{(0,0), (1,s), (2,2s), \ldots, (q-1, (q-1)s)\} \tag{6.1}$$

where $s \in \mathbb{Z}_q^n$.

For any two distinct binary strings $s_1, s_2 \in \mathbb{Z}_q^n$, $H(s_1)$ has only trivial intersection with $H(s_2)$. There are 2^n such binary strings. Now we use the Theorem 6.2 to establish the $\Omega(2^{n/2}) = \Omega(2^{0.5 \log^{-1} q \cdot \log |G|})$ query complexity. □

The high classical query complexity of the generalized injective hidden shift problem over group $\mathbb{Z}_m^n$ implies the high classical query complexity of the injective hidden shift problem over $\mathbb{Z}_m^n$:

Theorem 6.5. *The injective hidden shift problem with an underlying group of form $\mathbb{Z}_m^n$ has an exponential query complexity.*

Proof. Assuming we are given the generalized hidden shift problem, we can always ignore using the extra shifted oracles. So the query complexity of the hidden shift problem is at least equal to that of generalized hidden shift problem which by theorem 6.4 is exponential. □

Chapter 7

Concluding remarks and open problems

We offer some interesting questions and describe some of the future research directions in this chapter.

1. In chapter 3, we gave an efficient reduction from the non-injective hidden shift problem over any group to the injective hidden shift problem over the same group. One case of interest is when the underlying group is $\mathbb{Z}_q^n$. For the case when q is a constant prime power, efficient quantum algorithms are known to solve the injective hidden shift problem [Friedl et al., 2003][Ivanyos, 2008]. One interesting question which has been open for almost a decade now is whether an efficient quantum algorithm exists that can solve the injective hidden shift problem over $\mathbb{Z}_m^n$ when m is a composite number. Even for the smallest case ($m = 6$) this problem is still open. We know that the injective hidden shift problem over any abelian group A (like $\mathbb{Z}_m^n$) is equivalent to the non-abelian hidden subgroup problem when the underlying group is $A \rtimes \mathbb{Z}_2$ [Childs and van Dam, 2010]. On the other hand, we know that the quantum query complexity of the hidden subgroup problem is polynomial; and hence the query complexity of the hidden shift problem over $\mathbb{Z}_q^n$ is polynomial. However, we do not know of any time efficient quantum algorithm that solves this problem. The only partial progress is that we know how to solve this problem using subexponential amount of time, query and space based on the algorithm of [Kuperberg, 2005]. Subsequent improvements by [Regev, 2004] and [Childs et al., 2010] decreases the space complexity to polynomial.

2. Considering the hidden shift problem over $G = \mathbb{Z}_m^n$, we have an efficient quantum algorithm when m is a constant prime power. The algorithms are efficient in the sense that their query complexity and run-time complexity are polynomial in $\log(|G|) = n\log(m)$. Another extreme problem is when we keep $n = 1$ which is a small constant and let m grow. This problem is the hidden shift problem over $\mathbb{Z}_m$ which is equivalent to the dihedral hidden subgroup problem. As mentioned above, we have a subexponential time quantum algorithm for this problem. As it can be seen, there is quite a large gap between the run-time of the best known algorithms addressing these two extreme problems. We are interested in the complexity of the intermediate problems for which we let both m and n grow. It is interesting to know how the complexity of the intermediate problems between these two extremes evolve. This problem is also stated in [Ivanyos, 2008] and we believe that knowing how the complexity evolves can shed light on both the extreme cases as well.

3. In chapter 3, we gave a probabilistic lower bound on the probability that a random function can be injectivized successfully. It seems an interesting direction for the future research to characterize the set of functions that can be injectivized. This will be fruitful since the injectivization method seems to be a powerful tool since it does not put any restriction on the range of the functions and also the groups over which the functions are defined.

4. We showed in chapter 3 that the average case non-injective hidden shift problem is reducible to the injective hidden shift problem. It is interesting to know if the injective hidden shift problem can also be reduced to a class of the non-injective hidden shift problem which satisfy some plausible property that makes them tractable query-wise. This is a reasonable goal since we already know that because of the equivalence of the injective hidden shift problem to an instance of the non-abelian hidden subgroup problem, it can be solved using polynomial amount of queries.

5. In chapter 4, we gave a specific quantum algorithm for solving the hidden shift problem for c-almost generalized bent functions (which include the generalized bent functions). By the lower bounds on the query complexity of the unstructured search, it is well-known that an efficient algorithm for solving the worst case of the non-injective hidden shift problem is not feasible. However, by putting some constraint on the structure of the functions, the non-injective hidden shift problem can be made more tractable. As a future research direction, perhaps, there are many well-motivated functions for which it will be interesting to study the hidden shift problem. Two specific examples are the shifted Legendre symbols studied in [van Dam et al., 2003] and bent functions in [Rötteler, 2010] and [Gavinsky et al., 2011].

6. In the scheme we showed for solving the generalized hidden shift problem in chapter 5, at each step, our quantum algorithm is only using two shifted oracles and ignoring the rest. Does there exist an algorithm that uses fewer shifted oracles in total by using more than two shifted oracles at each step?

7. As it was discussed in chapter 2, in [Friedl et al., 2003], authors turn a set of linear inequations in $\mathbb{Z}_p$ into a set of polynomial equations. It is well known that solving a set of polynomial equations over any finite field is an NP-hard problem. However, this is not the case provided we are given a sufficiently overdefined set of equations. In fact, the authors in [Friedl et al., 2003] use the well-known classical technique of solving polynomial equations using the idea of linearization. It is crucial in their method to have an overdefined set of polynomial equation since by linearization, they form a set of linear equations but with more unknown variables as the overhead. Recently, there have been even more techniques discovered to cope with a polynomial set of equations. In this regard, there is a paper by Kipnis and Shamir [1999] in which they introduced a classical algorithm which is called *relinearization*. Unfortunately the time complexity of relinearization is not known. However, the belief is that its time complexity is just polynomial if we are given a set of polynomial equations which is overdefined to some small extent. There is an improved algorithm called XL introduced by Courtois, Klimov, Patarin, and Shamir [2000]. It is a simpler framework yet stronger than relinearization. As it is stated in [Courtois et al., 2000], they expect that upon being given $m \geq \epsilon n^2$ polynomial equations where n is the number of unknown variables and ϵ is any positive constant number less than 1/2, both of XL and relinearization have time complexity of approximately $n^{O(1/\sqrt{\epsilon})}$ which is just polynomial. Furthermore, they give strong evidence that both of these algorithms have subexponential time complexity when m only slightly exceeds n by a slowly growing number with n. Both of the papers [Kipnis and Shamir, 1999] and [Courtois et al., 2000] are highly cited in their respected field and have brought much excitement to the cryptography community. An improved algorithm over XL is called XSL and is suggested in [Courtois and Pieprzyk, 2002]. An interesting question is whether these methods can be used to solve the set of polynomial equations derived in [Friedl et al., 2003] and possibly to address the case of the hidden shift problem over $\mathbb{Z}_p^n$ when p is slowly growing with n by slightly modifying the methods. Recall that we only know how to solve the hidden shift problem when p is a constant and a prime power. An interesting and challenging direction for future research developing similar algorithms which can solve a set of linear inequations provided we are given sufficiently more linear inequations than the number of unknowns since this is usually the case with the sampling subroutines in the quantum algorithms that they

can produce an overdefined set of equations or inequations.

References

D. Gavinsky, M. Roetteler, and J. Roland. Quantum algorithm for the Boolean hidden shift problem. In *Proceedings of the 17th annual international conference on Computing and Combinatorics*, COCOON'11, pages 158–167, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22684-7. URL `http://dl.acm.org/citation.cfm?id=2033094.2033108`.

E. Bernstein and U. Vazirani. Quantum complexity theory. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing*, STOC '93, pages 11–20, New York, NY, USA, 1993. ACM. ISBN 0-89791-591-7. doi: http://doi.acm.org/10.1145/167088.167097. URL `http://doi.acm.org/10.1145/167088.167097`.

D. R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26: 116–123, 1994.

P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26:1484–1509, October 1997. ISSN 0097-5397. doi: 10.1137/S0097539795293172. URL `http://dl.acm.org/citation.cfm?id=264393.264406`.

W. van Dam, S. Hallgren, and L. Ip. Quantum algorithms for some hidden shift problems. In *Proceedings of the fourteenth annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 489–498, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. ISBN 0-89871-538-5. URL `http://dl.acm.org/citation.cfm?id=644108.644189`.

K. Friedl, G. Ivanyos, F. Magniez, M. Santha, and P. Sen. Hidden translation and orbit coset in quantum computing. In *In Proc. 35th ACM STOC*, pages 1–9. Press, 2003.

A. M. Childs and P. Wocjan. On the quantum hardness of solving isomorphism problems as nonabelian hidden shift problems. *Quantum Info. Comput.*, 7:504–521, July 2007. ISSN 1533-7146. URL `http://dl.acm.org/citation.cfm?id=2011832.2011838`.

R. Beals. Quantum computation of Fourier transforms over symmetric groups. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of Computing*, STOC '97, pages 48–53, New York, NY, USA, 1997. ACM. ISBN 0-89791-888-6. doi: http://doi.acm.org/10.1145/258533.258548. URL `http://doi.acm.org/10.1145/258533.258548`.

M. Ettinger and P. Høyer. A Quantum Observable for the Graph Isomorphism Problem. *eprint arXiv:quant-ph/9901029*, January 1999a.

P. Høyer. Efficient Quantum Transforms. *eprint arXiv:quant-ph/9702028*, February 1997.

D. Boneh and R. J. Lipton. Quantum cryptanalysis of hidden linear functions (extended abstract). In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '95, pages 424–437, London, UK, 1995. Springer-Verlag. ISBN 3-540-60221-6. URL `http://dl.acm.org/citation.cfm?id=646760.706007`.

O. S. Rothaus. On "bent" functions. *J. Comb. Theory, Ser. A*, pages 300–305, 1976.

C. Carlet and P. Gaborit. Hyper-bent functions and cyclic codes. *Journal of Combinatorial Theory, Series A*, 113(3):466 – 482, 2006. ISSN 0097-3165. doi: 10.1016/j.jcta.2005.04.008. URL `http://www.sciencedirect.com/science/article/pii/S0097316505000816`.

C. Carlet and A. Klapper. Upper bounds on the numbers of resilient functions and of bent functions. In *Proceedings of 23rd Symposium on Information Theory in the Benelux*, Louvain-La-Neuve, Belgium, 2002.

B. Preneel. *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.

P.V Kumar, R.A Scholtz, and L.R Welch. Generalized bent functions and their properties. *Journal of Combinatorial Theory, Series A*, 40(1):90 – 107, 1985. ISSN 0097-3165. doi: 10.1016/0097-3165(85)90049-4. URL `http://www.sciencedirect.com/science/article/pii/0097316585900494`.

K. Nyberg. Constructions of bent functions and difference sets. In *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '90, pages 151–160, New York, NY, USA, 1991. Springer-Verlag New York, Inc. ISBN 0-387-53587-X. URL `http://dl.acm.org/citation.cfm?id=112331.112344`.

M. Ettinger and P. Høyer. On quantum algorithms for noncommutative hidden subgroups. In *Proceedings of the 16th annual conference on Theoretical aspects of computer science*, STACS'99, pages 478–487, Berlin, Heidelberg, 1999b. Springer-Verlag. ISBN 3-540-65691-X. URL `http://dl.acm.org/citation.cfm?id=1764891.1764954`.

G. Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35:170–188, July 2005. ISSN 0097-5397. doi: http://dx.doi.org/10.1137/S0097539703436345. URL `http://dx.doi.org/10.1137/S0097539703436345`.

M. Rötteler. Quantum algorithms for highly non-linear boolean functions. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 448–457, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-98-6. URL `http://dl.acm.org/citation.cfm?id=1873601.1873638`.

M. Ozols, M. Roetteler, and J. Roland. Quantum rejection sampling. *ArXiv e-prints*, March 2011.

C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26:1510–1523, October 1997. ISSN 0097-5397. doi: 10.1137/S0097539796300933. URL `http://dl.acm.org/citation.cfm?id=264393.264407`.

G. Ivanyos. On solving systems of random linear disequations. *Quantum Info. Comput.*, 8:579–594, July 2008. ISSN 1533-7146. URL `http://dl.acm.org/citation.cfm?id=2016976.2016978`.

A. M. Childs and W. van Dam. Quantum algorithm for a generalized hidden shift problem. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1225–1232, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-24-5. URL `http://dl.acm.org/citation.cfm?id=1283383.1283515`.

Andrew M. Childs and Wim van Dam. Quantum algorithms for algebraic problems. *Rev. Mod. Phys.*, 82:1–52, Jan 2010. doi: 10.1103/RevModPhys.82.1. URL `http://link.aps.org/doi/10.1103/RevModPhys.82.1`.

J. Niel de Beaudrap, R. Cleve, and J. Watrous. Sharp Quantum vs. Classical Query Complexity Separations. *Algorithmica*, 34:449–461, 2002.

O. Regev. A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space. *eprint arXiv:quant-ph/0406151*, June 2004.

A. M. Childs, D. Jao, and V. Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *ArXiv e-prints*, December 2010.

A. Kipnis and A. Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In Michael Wiener, editor, *Advances in Cryptology CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 788–788. Springer Berlin / Heidelberg, 1999.

N. Courtois, E. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology, Eurocrypt2000, LNCS 1807*, pages 392–407. Springer-Verlag, 2000.

N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *ASIACRYPT*, pages 267–287, 2002.

G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum Amplitude Amplification and Estimation. *AMS Contemporary Mathematics*, 305:53–74, 2002.