University of Amsterdam

MSc Mathematics

Master Thesis

The Compressed Oracle and its Applications to Quantum Query Complexity

Author:
Sebastian Zur

Supervisors:
dr. C. Schaffner
dr. C. Majenz
J. Czajkowski, MSc

Examination date:
July 17, 2019

Daily supervisor:
J. Czajkowski, MSc

Korteweg-de Vries Institute for Mathematics

QuSoft
Research Center for Quantum Software

Abstract

We propose a new framework for lower bounding the quantum query complexity of certain problems by viewing these problems as relations. We combine this framework with the new compressed-oracle technique by Zhandry [Zha18], which we first formalize and construct a unitary implementation for. This implementation is given in both an high-level pseudocode, as well as a unitary algorithm in ProjectQ. Based on recent work by [LZ18] the application of the compressed-oracle technique to our new framework allows for a general method to find lower bounds on the quantum query complexities.
Our results improve upon the current known quantum query lower bound of finding k $r-$collisions, extending the result of $\Omega\left(k^{2/(r+1)}N^{1/(r+1)}\right)$ for $k \leq N^{1/2^r}$ to hold for $k \leq \sqrt{\frac{N}{r}}$. We also present new lower bounds in the case of parallel queries for the Collision, K-Sum, Chain of Values and Multiclaw problem.

Title: The Compressed Oracle and its Applications to Quantum Query Complexity
Author: Sebastian Zur, zursebastian@gmail.com, 10547800
Supervisors: dr. C. Schaffner, dr. C. Majenz, J. Czajkowski, MSc
Daily supervisor: J. Czajkowski, MSc
Second Examiner: dr. M. Walter
Examination date: July 17, 2019

Korteweg-de Vries Institute for Mathematics
University of Amsterdam
Science Park 105-107, 1098 XG Amsterdam
http://kdvi.uva.nl

QuSoft
Science Park 123, 1098 XG Amsterdam
http://www.qusoft.org/

Acknowledgements

First and foremost I would like to thank my supervisors, Christian Schaffner, Christian Majenz and Jan Czajkowski, for their contribution to this thesis. They have spent many hours reading and reviewing my work, introducing me to possible subjects to tackle and allowed me to contribute to a publication. Most importantly they, together with the rest of the QuSoft crypto group, created a welcoming and fun environment to finish this final part of my master's programme. Christian Schaffner was also the one to welcome me to the field of quantum cryptography four years ago and since then has always been happy to meet up and and be an helping hand throughout those years, something for which I am very grateful.

I would like to thank Jan especially for acting as my daily supervisor, allowing me to drop by and bother him at every moment of the day, whether it be at his office, mail or by phone. His investment in deciphering my incoherent writings, providing elaborate feedback, preparing all of our meetings and explaining new concepts has helped me tremendously throughout this whole project.

Huge gratitude also goes out to my family (the old and new one) for always allowing me to prioritize my studies, feeding me (which is not light work) and never doubting me. The decision of my parents to motivate the younger (and lazier) me into combining mathematics with computer science has shaped my life for the better. Explicit thanks go out to Chelsea, who has put me on a pedestal these last few stressful weeks, making her a much larger part of this work than she might realise.

Lastly I would like to thank Michael Walter and Arno Kret for taking the time to review this thesis in their respective roles of second examiner and representative of the examination board.

Contents

List of Symbols

$\mathrm{Tr}[A]$	Trace of a linear operator A 11
A^*	Conjugate transpose of a linear operator A 11
$\mathbb{C}^d$	Complex coordinate space of dimension d 11
$\|\|\|\psi\rangle\|\|$	Norm of a vector $\|\psi\rangle$ 11
$\mathbb{I}$	Identity matrix 11
$\|x\|$	Cardinality of a set x/norm of a complex number x 11
$\mathbb{Z}_2$	Cyclic group of order 2 11
$\oplus$	Bitwise XOR 11
$\mathcal{O}$	Landau complexity class 11
Ω	Landau complexity class 11
$\{\|0\rangle, \|1\rangle\}$	Computational basis 11
$\|\psi\rangle$	Pure quantum state 12
$\mathcal{H}$	Hilbert space 12
ρ	Density matrix/mixed quantum state 12
QFT_N	Quantum Fourier Transform 14
ROM	Random-oracle model 15
QROM	Quantum random-oracle model 15
D	Database in the standard basis/database register 16
StO	Standard oracle 16
PhO	Phase oracle 17
FO	Fourier oracle 17
$\mathsf{P}_{x,\eta}$	Point function 17
Δ	Compressed database in the Fourier basis 17
CFO	Compressed Fourier oracle 18
$\mathsf{F}\Delta$	Database in the Fourier basis 19
Dec	Decompressing algorithm 20
Add	Subroutine to add query to the database 20
Upd	Subroutine to update query in the database 20
Rem	Subroutine to remove query from the database 20
CStO	Compressed standard oracle 21
CPhO	Compressed phase oracle 21
R^r	Relation 26
$\mathsf{R}^r_{\mathsf{Col}}$	Collision relation 26
$\mathsf{R}^r_{\mathsf{Sum}}$	Sum relation 28
$P_{r,k}$	Projector onto the database containing k distinct r-tuples satisfying some relation 28

$G_{\mathsf{R}^r,t}$	Set of completion values for $(r-t)$-tuples for R^r 35
$\mathsf{R}^r_{\mathsf{CoV}}$	Chain of Values relation 59
$\mathsf{R}^r_{\mathsf{Mc}}$	Multiclaw relation 60
Locate	Subroutine to locate position of x in the database 68
Clean	Subroutine to clean up auxiliary registers 68
Larger	Subroutine to compare two bit strings 68
P	Permutation unitary 68

Introduction

Since the idea of a quantum computer has been suggested by [Fey82], the academic world has been getting closer to building a quantum computer capable of computations beyond that of its classical counterpart. Whereas this broadens the capabilities of our society to model the world around us [BC98], it also increases the power of potential adversaries in the context of cryptography.

Since it has been shown that under the presence of a quantum computer RSA-based cryptographic schemes are no longer secure [Sho94], many quantum cryptographers have been trying and succeeding in proving the security of existing schemes under the presence of adversaries with quantum capabilities [Bon+11; Cza+18; TU16; Unr17]. Many of these use the quantum random-oracle model (QROM, see Section 2.1) as a proving tool, just as the random-oracle model is used in classical security proofs. None of the existing QROM techniques however were capable of recording queries done by the adversary, up until this problem was recently resolved by Zhandry [Zha18]. This technique is aptly named the compressed-oracle technique, as will be shown in Section 2.3.

This new technique has immediately resulted in numerous security proofs of schemes in the presence of quantum adversaries, where in the classical proofs some recording of the adversary's query was necessary [Cza+19; HI19; LZ19; Zha18]. In these proofs the compressed-oracle technique is used to upper bound the success probability of the adversary distinguishing between a simulator and the real case [Cor+05; MRH04]. This new method also serves as an alternative to the already existing methods of finding lower bounds on the quantum query complexity of complexity problems like the adversary method [Amb02] and polynomial method [Bea+01].

This work

In this thesis we build forth on the innovating ideas as presented by Zhandry [Zha18]. We use these as blueprints to create and program a quantum algorithm that simulates a random oracle and is indistinguishable from a random oracle to any adversary interacting with this algorithm. The algorithm is a specific instance of the more general result by Czajkowski, Majenz, Schaffner and Zur [Cza+19], where an algorithm is given to quantum lazy sample from any function distribution, instead of just the uniform distribution. We present this algorithm in a high level (Algorithm 1), using smaller subroutines instead of explicit gates. As a formal check this implementation is then programmed by making use of only unitary operations [Zur19] into ProjectQ [HST17].

The main result of this thesis is a generalisation of the recent result by [LZ18], where the compressed-oracle technique is used to give a tight bound (an algorithm matching this bound is also given in the same work) on the quantum query complexity of the Collision problem. In this problem an adversary is tasked with finding r different inputs of a given function $F : \{0,1\}^m \to \{0,1\}^n$ which map to the same output value. This problem holds great significance in the context of cryptography when analyzing hash functions [CN08; Din+14; Nai+13]. We have managed to improve on the quantum query lower bound of finding k $r-$collisions, extending the result of $\Omega\left(k^{2/(r+1)}N^{1/(r+1)}\right)$ for $k \leq N^{1/2^r}$ to hold for $k \leq \sqrt{\frac{N}{r}}$. This technique has then been generalised, such that it can be applied to lower bound the quantum query complexity of various problems:

- The K-Sum problem (Corollary 3.4.4), where an adversary is tasked with finding r output values summing to zero (viewed over $\mathbb{Z}_{2^n}$).
- The Chain of Values problem (Corollary 5.1.2), where an adversary is tasked with finding a sequence $x, F(x), F(F(x)), ..., F^{r-1}(x)$.
- The Multiclaw problem (Corollary 5.2.2), where an adversary is tasked with finding a input value for each function $F_i : \{0,1\}^{m_i} \to \{0,1\}^n$ with $i \in [r]$ such that all these input values map to the same output value.

To derive these results, we have constructed a framework for computational-complexity problems by modelling them as *relations*. We divide these relations into two categories; *iterative relations* (Chapter 3) and *non-iterative relations* (Chapter 4). The intuitive difference between these two categories is that iterative relations are created from smaller relations. An example would be the Collision problem, where every $r-$collisions is also a $(r-1)-$collision. For both of these categories an upper bound is given on the quantum query solvability, i.e. the probability of success as a function of the number of queries. The lower bound on the quantum query complexity can then be derived from this upper bound on the quantum query solvability. To illuminate on this technique we first start with a specific problem of both classes before lastly generalising the technique.

Relatively little research has been done on quantum parallel query complexity [JMW17]. In this setting the adversary is allowed to query multiple non-adaptive queries with every query step. The versatility of the compressed-oracle technique also proves itself here, as it has allowed us to derive bounds on the quantum query complexity of non-iterative relations in this setting. For iterative relations this also succeeded in the case of the Collision problem, but for general iterative relations we have only managed to bound the quantum query complexity in the event of a single fully parallel query. This fully parallel result (Corollary 4.4.1) showcases that we need sequential queries to achieve a speedup over the classical case.

For a summary of our quantum query complexity lower bounds see Section 6.2.

Overview

We start the thesis by providing a small introduction into the subject of quantum computing and introducing the necessary preliminaries, definitions, and notation necessary to understand the rest of the thesis. In Chapter 2 we start with further elaborating on the motivation behind Zhandry's compressed-oracle technique, before introducing and explaining the technique itself. From this we derive the algorithm and also discuss the programmed implementation. In Chapter 3 we make the leap to quantum query solvabililty, where we introduce the notion of (iterative/non-iterative) relations. After demonstrating the compressed-oracle technique on the K-sum problem we move on to generalising/parallelising the results for any non-iterative relation. In Chapter 4 we extend the technique based on [LZ18] and then generalise/parallelise the corresponding result. The thesis is then concluded with some extra applications of the proven theorems to either create new or improve upon existing quantum query complexity results.

1. Quantum Computation

This chapter will only be a brief introduction to quantum computing. For a more in depth introduction we recommend [Wol11] or (if there is a need for either more thorough background or some light reading matter) [NC02].

1.1. Notation

We will assume basic familiarity with linear algebra and probability theory. The trace of a linear operator A is denoted by $\mathrm{Tr}[A]$ and its conjugate transpose as A^*. Throughout this thesis we will work on complex Hilbert spaces $\mathbb{C}^d$ of some finite dimension d, equipped with the canonical Hermitian inner product. This inner product induces a norm on vectors $|\psi\rangle \in \mathbb{C}^d$, which we denote by $\||\psi\rangle\| = \sqrt{\mathrm{Tr}[|\psi\rangle\langle\psi|]}$. For the identity matrix we shall write $\mathbb{1}$. $|x|$ is used interchangeably to denote either the cardinality of a set x or the norm of a complex number x.
The set of n-bit strings is denoted by $\{0,1\}^n$. Mathematically this set is equivalent to $\mathbb{Z}_2^n$, where $\mathbb{Z}_2$ is the cyclic group of order 2. The group operation on these n-bit strings is the bitwise XOR, denoted by $x \oplus y$, which corresponds to the coordinate-wise addition on $\mathbb{Z}_2^n$. For the bitwise inner product of two n-bit strings we write $x \cdot y$. Here the usage of the term 'inner product' is quite footloose, as we will not be viewing $\mathbb{Z}_2^n$ as the vector space $\mathbb{F}_2^n$. At rare occassions $\cdot$ will also be used to denote the multiplication operator whenever this is required for readability. If we refer to the i-th bit of $x \in \{0,1\}^n$, we denote this with the subscript x_i. We will abbreviate $|\{0,1\}^n| = 2^n = N$ and $|\{0,1\}^m| = 2^m = M$.
The following two functions will be used in this thesis:

- The indicator function $\mathbb{1}_{a\leq b} := \begin{cases} 1, & \text{if } a \leq b \\ 0, & \text{else} \end{cases}$,
- The Kronecker delta function $\delta_{a,b} := \begin{cases} 1, & \text{if } a = b \\ 0, & \text{else} \end{cases}$.

Lastly we use the following notation from complexity theory: for f a complex function and g a positive function, we write $f(x) = \mathcal{O}(g(x))$ if $\limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty$ and $f(x) = \Omega(g(x))$ if $\limsup_{x\to\infty} \left|\frac{g(x)}{f(x)}\right| < \infty$. All other notation will be introduced throughout this thesis.

1.2. Quantum states

1.2.1. Qubits

Where in classical computer science the basic unit of information is a bit, in quantum computing this unit is the *qubit*, which is represented by a unit vector in $\mathbb{C}^2$. The standard basis on $\mathbb{C}^2$ is referred to as the *computational basis* $\{|0\rangle, |1\rangle\}$, where

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1.1}$$

represent the classical bits. Here we have written the vectors in *Dirac notation*. We thus see from the above definition that any qubit can be written as

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \tag{1.2}$$

with $\alpha_0, \alpha_1 \in \mathbb{C}$ and $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Here $|\psi\rangle$ is said to be a *superposition* of the classical states $|0\rangle, |1\rangle$.
Multiple qubits can be combined by taking the tensor product over the individual qubits:

$$|\psi\phi\rangle_{XY} := |\psi\rangle_X \otimes |\phi\rangle_Y \in \left(\mathbb{C}^2\right)^{\otimes 2}. \tag{1.3}$$

The registers (denoted by the subscript) clarify to which Hilbert space each qubit belongs. We will omit these registers whenever this is clear from the context. It follows that n-qubit states live in the 2^n-dimensional Hilbert space $(\mathbb{C}^2)^{\otimes n}$, spanned by the computational basis consisting of the n-bit strings. In general the underlying Hilbert space in quantum mechanics does not need to be $(\mathbb{C}^2)^{\otimes n}$. For $|\psi\rangle$ living in any such general Hilbert space $\mathcal{H}$ we refer to $|\psi\rangle$ as a quantum state instead of an n-qubit state.

1.2.2. Pure and mixed states

Another way of describing quantum states is by density matrices. These are positive semi-definite Hermitian operators with trace 1. Every quantum state can be assigned a density matrix as follows: $\rho := |\psi\rangle\langle\psi|$. Since Hermitian operators form a group under addition (and using the linearity of the trace), we see that any convex combination of density matrices again results in a density matrix $\rho = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$. Whenever there are at least two $\lambda_i > 0$, this state can not be written as $|\phi\rangle\langle\phi|$ and thus is not pure. These density matrices will be referred to as *mixed states*. Such a mixed state can been thought of as an ensemble of pure states $\{|\psi_i\rangle\}$, each occurring with probability λ_i.
We can always transform a mixed state into a pure state, although this comes at the expensive of adding an extra register. To do this, we first need to introduce the partial trace. For any two Hilbert spaces $\mathcal{H}_X, \mathcal{H}_Y$ the partial trace over $\mathcal{H}_X$ is defined as the unique operator Tr_X such that

$$\forall \rho_X \in \mathcal{H}_X, \sigma_Y \in \mathcal{H}_Y : \quad \mathrm{Tr}_X\left[\rho_X \otimes \sigma_Y\right] := \mathrm{Tr}[\rho_X]\sigma_Y. \tag{1.4}$$

Now by Theorem 4.1 from [Wat18] for every ρ_X there exists a *purification* $|\psi\rangle_{XY}$ such that

$$\mathrm{Tr}_Y\left[|\psi\rangle\langle\psi|_{XY}\right] = \rho_X \tag{1.5}$$

if and only if $\dim(\mathcal{H}_Y) \geq \dim(\rho_X)$.

1.3. Quantum operations

1.3.1. Gates

Quantum algorithms can manipulate qubits (or quantum states in general), just as classical algorithms manipulate bits. When transforming a quantum state, we want the result to still satisfy the definition of what a quantum state entails; a unit vector over some Hilbert space. So any such transformation should correspond to a norm-preserving linear endomorphism, i.e. a unitary matrix U, called a *gate*. Since unitary matrices are invertible, this implies that any gate is reversible, which does not always hold in the classical case. The gate U is applied to a quantum state by

$$U : |\psi\rangle \to U|\psi\rangle, \qquad U : \rho \to U\rho U^*. \tag{1.6}$$

If we wish to emphasize on which particular register the gate is applied, we add a superscript to the gate:

$$\left(\mathbb{I}^X \otimes U^Y\right) |\psi\rangle_{XY}. \tag{1.7}$$

If the underlying Hilbert space of register X is an n-fold tensor product of a smaller space, for example $(\mathbb{C}^2)^{\otimes n}$, and we wish to apply U coordinate-wise, we abbreviate the total gate $U^{\otimes n} = U^X$.

An example of a gate that we will use is the *Hadamard* transform H. When describing a gate, one can either give the matrix representation or describe how the gate works on basis states:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad \begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \\ H|1\rangle &= \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle. \end{aligned} \tag{1.8}$$

1.3.2. Measurements

Instead of only applying the above reversible operations, one can also *measure* a quantum state. This is done by a measurement; a countable set of positive semi-definite matrices $\{P_i\}_i$ that sum to the identity. The probability of an outcome i is the squared norm of the resulting state, i.e. $\|P_i|\psi\rangle\|^2 = \mathrm{Tr}[P_i|\psi\rangle\langle\psi|]$ (or $\|P_i\rho P_i\|^2 = \mathrm{Tr}[P_i\rho]$ when working with density matrices). Since the resulting state $P_i|\psi\rangle$ is not always necessarily a unit vector, the convention is to normalize the result. In this thesis we will restrict ourselves to *projective measurements*, where the P_i have the extra property being pairwise orthogonal projections. As a result the respective spaces that the P_i project onto will also be an orthogonal, a property that we will make plenty of use of in this work to establish equality in the Cauchy–Schwarz inequality.

A canonical example of a projective measurement is the measurement in the computational basis: $\{|i\rangle\langle i|\}_{i\in\{0,1\}^n}$. This measurement projects any n-qubit state $|\psi\rangle = \sum_{i\in\{0,1\}^n} \alpha_i|i\rangle$ onto the classical n-bit string i with probability $|\alpha_i|^2$

1.4. Quantum Fourier transform

Quantum algorithms consist of the previously mentioned gates and measurements. By exploiting superposition some of these algorithms manage to significantly outperform their classical counterparts [Sho94; Gro96]. One of these algorithms that we will make use of in this thesis is the *Quantum Fourier Transform*. Classically the discrete Fourier Transform [CT65] has many applications, ranging from signal-processing to complexity theory. The discrete Fourier Transform F_N is defined as the following $N \times N$ matrix

$$F_N = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{-1} & \omega_N^{-2} & \cdots & \omega_N \end{pmatrix}, \tag{1.9}$$

where $\omega_N = e^{2\pi i/N}$ is the N-th root of unity. The inner product of any two columns k, l results in

$$\frac{1}{N}\sum_{j=0}^{N-1}\left(\omega_N^{jk}\right)^* \omega_N^{jl} = \frac{1}{N}\sum_{j=0}^{N-1}\omega_N^{j(l-k)} = \delta_{k,l} \tag{1.10}$$

and thus F_N is a unitary matrix. This equality will come to our assistance often throughout the thesis. As a result of F_N being a unitary matrix, it can be applied to any quantum state living on a Hilbert space of dimension N. In this quantum setting F_N is instead referred to as QFT_N. In this work we will always consider $N = 2$, but the general case is needed when we wish to quantum lazy sample more general distributions [Cza+19]. We abbreviate this QFT_2 by QFT:

$$\mathsf{QFT}|y\rangle := H^{\otimes n}|y\rangle = \frac{1}{\sqrt{2^n}}\sum_{\eta\in\{0,1\}^n}(-1)^{\eta\cdot y}|\eta\rangle. \tag{1.11}$$

The resulting image of the computational basis under QFT will be referred to as the *Fourier basis*.

2. Compressed Oracles

2.1. Random-oracle model

2.1.1. Classical

Cryptographic hash functions are constructed to be as close to a deterministic uniform random function as possible. We settle on 'as close to' because creating an actually efficient deterministic random function is infeasible. To guarantee the randomness the output needs to be sampled uniformly random. Combining this with the deterministic nature however would require some sort of lookup-table to maintain consistency in the output. By adding efficiency to this stew of properties we end up at a contradiction, as any such lookup-table would be exponential in the input size. A workaround is lazy-sampling, where an entry is only added to the lookup-table whenever an input value is queried for the first time by the adversary. This guarantees that the lookup-table will be polynomial in size, assuming that the adversary is only allowed to make polynomially many queries.

In the field of cryptography, it sometimes turns out to be difficult to find a proper and rigorous mathematical proof of the security of some cryptographic system. One devised tool to ease this process is the random-oracle model (ROM) [BR93]. In the ROM, every hash function is replaced by a random oracle, a theoretical black box that generates output precisely in the deterministic and uniform random way that we would like it to. If the system is proven to be secure under this substitution, one has proven the system to be secure in the ROM. Even though this does not provide absolute certainty that the original scheme was secure, it is widely believed that security in the ROM provides practical security in the real world [BR93].

In practice many schemes [Ber+07; Dam89; Mer89] implement hash functions constructed out of smaller functions, which are called internal functions. When these internal functions are public, it is not hard for any adversary to see that the hash construction itself is not a random function. It has been suggested in [MRH04] that the ROM is still an excellent tool in proving the security of these constructions, by instead modelling the internal functions as random oracles.

2.1.2. Quantum

It would be convenient if our ROM extended to the quantum setting. This would allow the ROM to be used to proof security in the case of a quantum adversary that can query a su-

perposition of input values to the given hash function. The model where the adversary is allowed to make queries in superposition to the random oracle model is called the quantum random-oracle model (QROM) [Bon+11]. This does set the stage for a potential problem: how should the random oracle remain deterministic without using an exponential amount of memory if an adversary can query a superposition over exponentially many input values? A workaround for this problem has been created by not storing any information about the queries done by the adversary, called history-free reductions [Bon+11], but this method fails in the case of proving security hash functions containing internal functions.

2.2. Oracle variations

In [Zha18] a simple, but nevertheless elegant solution is given to the aforementioned problem of recording the queries of the adversary. We will show different, but equivalent interpretations of an oracle for a function $D : \{0,1\}^m \to \{0,1\}^n$. The truth table of this function can be represented as a vector in $\left(\mathbb{C}^{2^m}\right)^{\otimes 2^n}$ where the x-th entry of D contains the n-bit string $D(x)$. This vector can be seen as a database, which is why the letter D has been chosen.

2.2.1. Standard and phase oracle

The two most canonical oracles are the standard and phase oracle. The standard oracle StO works as follows on any joint basis state of the query in the database:

$$\mathsf{StO}\left(|x,y\rangle_{XY} \otimes |D\rangle_D\right) = |x, y \oplus D(x)\rangle_{XY} \otimes |D\rangle_D. \tag{2.1}$$

Here the XY registers belong to the adversary, whereas D is inaccessible for the adversary. Since in the QROM D is a random function, we want the D register to be a uniform probability distribution over all possible truth tables. We have seen in Section 1.2.2 that this uniform probability distribution can be realised with mixed states.

$$\begin{aligned}&\mathsf{StO}\left(\frac{1}{2^{n2^m}}\sum_D |x,y\rangle\langle x,y|_{XY} \otimes |D\rangle\langle D|_D\right)\mathsf{StO}^* \\ &= \frac{1}{2^{n2^m}}\sum_D |x, y\oplus D(x)\rangle\langle x, y\oplus D(x)|_{XY} \otimes |D\rangle\langle D|_D\,.\end{aligned} \tag{2.2}$$

By tracing out the D register, we find that the adversary's register after the query is equal to

$$\frac{1}{2^{n2^m}}\sum_D |x, y\oplus D(x)\rangle\langle x, y\oplus D(x)|_{XY}\,. \tag{2.3}$$

Had we instead opted to initialize D to the uniform superposition over all truth tables, which is a purification of D being mixed over all possible truth tables, by Equation 2.1 the state after the query is equal to

$$\frac{1}{\sqrt{2^{n2^m}}}\sum_D |x, y\oplus D(x)\rangle_{XY}. \tag{2.4}$$

So both interpretations of the oracle result in the same resulting state on the adversary's side. Since working with pure states is in general easier than with mixed states, we will use this purification approach.

A well known variation of the StO oracle implementation is the phase oracle (PhO). Here the output of D is provided in the Fourier basis, by applying QFT to the Y-register of the adversary. For clarity we will use Greek letters to emphasize whenever we are working in the Fourier basis (as opposed to the regular basis).

$$\mathsf{PhO} = (\mathbb{I}^X \otimes \mathsf{QFT}^Y \otimes \mathbb{I}^D)\mathsf{StO}(\mathbb{I}^X \otimes \mathsf{QFT}^Y \otimes \mathbb{I}^D). \tag{2.5}$$

Lemma 2.2.1. *Let* StO *be defined as in Equation 2.1 and* PhO *as in Equation 2.5. Then for any query to* PhO *of the form* $|x,\eta\rangle_{XY} \otimes |D\rangle_D$ *it holds that*

$$\mathsf{PhO}\left(|x,\eta\rangle_{XY} \otimes |D\rangle_D\right) = (-1)^{\eta \cdot D(x)}|x,\eta\rangle_{XY} \otimes |D\rangle_D. \tag{2.6}$$

Proof. Writing out the definitions of StO and PhO results in

$$\begin{aligned}
\mathsf{PhO}\left(|x,\eta\rangle_{XY} \otimes |D\rangle_D\right) &= (\mathbb{I}^X \otimes \mathsf{QFT}^Y \otimes \mathbb{I}^D)\mathsf{StO}(\mathbb{I}^X \otimes \mathsf{QFT}^Y \otimes \mathbb{I}^D)\left(|x,\eta\rangle_{XY} \otimes |D\rangle_D\right) \\
&= (\mathbb{I}^X \otimes \mathsf{QFT}^Y \otimes \mathbb{I}^D)\mathsf{StO}\left(\frac{1}{\sqrt{2^n}}\sum_y (-1)^{y\cdot\eta}|x,y\rangle_{XY} \otimes |D\rangle_D\right) \\
&= (\mathbb{I}^X \otimes \mathsf{QFT}^Y \otimes \mathbb{I}^D)\left(\frac{1}{\sqrt{2^n}}\sum_y (-1)^{y\cdot\eta}|x,y\oplus D(x)\rangle_{XY} \otimes |D\rangle_D\right) \\
&= \frac{1}{2^n}\sum_{y,\omega}(-1)^{\omega\cdot(y\oplus D(x))}(-1)^{y\cdot\eta}|x,\omega\rangle_{XY} \otimes |D\rangle_D \\
&= \frac{1}{2^n}\sum_{y,\omega}(-1)^{y\cdot(\omega\oplus\eta)}(-1)^{\omega\cdot D(x)}|x,\omega\rangle_{XY} \otimes |D\rangle_D \\
&= \sum_{\omega}\delta_{\omega,\eta}(-1)^{\omega\cdot D(x)}|x,\omega\rangle_{XY} \otimes |D\rangle_D \\
&= (-1)^{\eta\cdot D(x)}|x,\eta\rangle_{XY} \otimes |D\rangle_D,
\end{aligned} \tag{2.7}$$

where in the second-to-last equality we have used the n-fold tensor product of Equation 1.10. □

2.2.2. Fourier oracle

This change of basis we just applied to the Y-register can also be applied to the D-register. We will call the resulting oracle operation the Fourier oracle (FO).

$$\mathsf{FO} = (\mathbb{I}^X \otimes \mathbb{I}^Y \otimes \mathsf{QFT}^D)\mathsf{PhO}(\mathbb{I}^X \otimes \mathbb{I}^Y \otimes \mathsf{QFT}^D). \tag{2.8}$$

As we initialized D to the uniform superposition over all D, this corresponds to 0 in the Fourier basis. Before we show the usefulness of this extra basis transformation, we will first show how this operator looks in more detail, just as we did with PhO. Let $\mathsf{P}_{x,\eta}$ denote the point function where $\mathsf{P}_{x,\eta}(x') = \eta \cdot \delta_{x,x'}$, then the following holds:

Lemma 2.2.2. *Let* PhO *be defined as in Equation 2.5 and* FO *as in Equation 2.8. Then for any query to* FO *of the form* $|x,\eta\rangle_{XY} \otimes |\Delta\rangle_D$ *it holds that*

$$\mathsf{FO}(|x,\eta\rangle_{XY} \otimes |\Delta_D\rangle) = |x,\eta\rangle_{XY} \otimes |\Delta \oplus \mathsf{P}_{x,\eta}\rangle_D, \tag{2.9}$$

where $\mathsf{P}_{x,\eta}(x') = \eta \cdot \delta_{x,x'}$ *is the point function.*

Proof. Writing out the definitions of StO and PhO in combination with Lemma 2.2.1 results in

$$\begin{aligned}
\mathsf{FO}(|x,\eta\rangle_{XY} \otimes |\Delta\rangle_D) &= (\mathbb{I}^X \otimes \mathbb{I}^Y \otimes \mathsf{QFT}^D)\mathsf{PhO}(\mathbb{I}^X \otimes \mathbb{I}^Y \otimes \mathsf{QFT}^D)(|x,\eta\rangle_{XY} \otimes |\Delta\rangle_D) \\
&= (\mathbb{I}^X \otimes \mathbb{I}^Y \otimes \mathsf{QFT}^D)\mathsf{PhO}\left(\frac{1}{\sqrt{2^{n2^m}}}\sum_D (-1)^{D\cdot\Delta}|x,\eta\rangle_{XY} \otimes |D\rangle_D\right) \\
&= (\mathbb{I}^X \otimes \mathbb{I}^Y \otimes \mathsf{QFT}^D)\left(\frac{1}{\sqrt{2^{n2^m}}}\sum_D (-1)^{D\cdot\Delta}(-1)^{\eta\cdot D(x)}|x,\eta\rangle_{XY} \otimes |D\rangle_D\right) \\
&= \frac{1}{2^{n2^m}}\sum_{D,\Omega}(-1)^{\Omega\cdot D}(-1)^{D\cdot\Delta}(-1)^{\eta\cdot D(x)}|x,\eta\rangle_{XY} \otimes |\Omega\rangle_D \\
&= \frac{1}{2^{n2^m}}\sum_{D,\Omega}(-1)^{D\cdot(\Omega\oplus\Delta\oplus\mathsf{P}_{x,\eta})}|x,\eta\rangle_{XY} \otimes |\Omega\rangle_D \\
&= \sum_{\Omega}\delta_{\Omega,\Delta\oplus\mathsf{P}_{x,\eta}}|x,\eta\rangle_{XY} \otimes |\Omega\rangle_D \\
&= |x,\eta\rangle_{XY} \otimes |\Delta \oplus \mathsf{P}_{x,\eta}\rangle_D,
\end{aligned} \tag{2.10}$$

where in the second-to-last equality we have used the $m2^n$-fold tensor product of Equation 1.10. □

2.3. Compressed Oracle

Using Lemma 2.2.2 we can show the ingenuity of looking at the oracle from this specific perspective. As we have seen in the classical case, after q queries by the adversary, our database contains at most q entries. As we in practice assume the adversary to make at most polynomially many queries, we thus are required to save a polynomially sized database. In the quantum setting however the size of the database would have to be exponential in the input size of the query, because the adversary can query the superposition over all input values in a single query.

When we look at FO however, we notice that after q queries the resulting database Δ is the sum of at most q point functions, since the initialization of Δ was the all-zero vector. As every point function $\mathsf{P}_{x,\eta}$ can be represented by a tuple (x,η), we can actually compress Δ to a list of at most q tuples (x,η). We will call this compressed-oracle variant the compressed Fourier oracle (CFO). This operation acts by

$$\mathsf{CFO}(|x,\eta\rangle_{XY} \otimes |\Delta\rangle_D) = |x,\eta\rangle_{XY} \otimes |\Delta \oplus (x,\eta)\rangle_D. \tag{2.11}$$

By looking at Lemma 2.2.2 and the point function, we derive that this $\Delta \oplus (x, \eta)$ operation has the following properties: $\forall x, x' \in \{0,1\}^m$ and $\forall \eta, \eta' \in \{0,1\}^n$ it holds that

- $\mathsf{P}_{x,0}(x') = 0$. Thus $\Delta \oplus (x, 0)$ should return Δ.
- $\mathsf{P}_{x,\eta}(x') + \mathsf{P}_{x,\eta'}(x') = (\eta \oplus \eta') \cdot \delta_{x,x'} = \mathsf{P}_{x,(\eta\oplus\eta')}(x')$. So if there is already a pair (x, η) for any $\eta \neq \eta'$ in Δ, then $\Delta \oplus (x, \eta')$ should replace (x, η) with $(x, \eta \oplus \eta')$ in Δ and return the updated Δ.
- If we look at the previous case whenever $\eta = \eta'$, we find $\mathsf{P}_{x,\eta}(x') + \mathsf{P}_{x,\eta'}(x') = 0$. Thus in this case (x, η) is removed from Δ and the updated Δ is returned.
- Conversely if there is no (x, η) for any η in Δ, (x, η) is added to Δ and the updated Δ is returned.

By linearity it follows how CFO acts on any quantum state. There remains the question of how we order the list of tuples in our compressed database. Since FO stays invariant under permutations on the order of the queries, so should CFO. As such we decide to order the tuples by x value, from smallest to largest. It is clear from the properties of CFO that Δ never contains more two tuples with the same x value, so this ordering is unambiguous. From here on Δ will refer to the compressed version, while we will denote the original truth table of D in the Fourier basis by $\mathsf{F}\Delta$.

For proof purposes we will use the high level implementation of CFO, as described in Algorithm 1. Both the detailed and high level algorithms as stated here are specific instances of the algorithms described in [Cza+19], where a general CFO is given that samples for any given distribution instead of just the uniform distribution. In this more general implementation QFT_N is used instead of QFT_2.

Algorithm 1: High level CFO

Input : Adversary query and database: $|x, \eta\rangle_{XY}|\Delta\rangle_D$
Output: $|x, \eta\rangle_{XY}|\Delta'\rangle_D$

```
if x ∉ Δ^X then                                   // add entry to the database
    Add x to Δ^X in the right place               // keep Δ^X sorted properly
XOR η into Δ^Y(x)                                 // update entry with x
for i = 1, 2, ..., q do
    if Δ_i^Y = 0 then                             // remove faulty entries
        Remove x from Δ_i^X
return |x, η⟩_XY |Δ'⟩_D                           // Δ' is the modified database
```

We have decided to keep CFO a unitary operation and as such the size of Δ is fixed to have space for precisely q queries. Any extra space is implemented as empty registers, which are padded behind the entries of Δ. One can choose to change the functioning CFO by allowing the creation and removal of registers, which as a result turns CFO into an isometry.

As we claim equivalence between CFO and FO, there should be some invertible Dec such that the following holds:

$$\mathsf{Dec} \circ \mathsf{CFO} = \mathsf{FO} \circ \mathsf{Dec}. \tag{2.12}$$

The construction of such a decompressing algorithm is actually surprisingly simple, as described in Algorithm 2. Just as with CFO, one can choose to change the implementation of

Algorithm 2: Dec

Input : Empty Fourier database and compressed Fourier database: $|\mathsf{F}\Delta\rangle_{\mathsf{F}\Delta}|\Delta\rangle_D$
Output: $|\mathsf{F}\Delta'\rangle_{\mathsf{F}\Delta}|\Delta'\rangle_D$

1 Set $|a\rangle_A = |0 \in [q]\rangle_A$ // initialize auxiliary register A
2 **for** $i = 1, ..., q$ **do**
3 $\quad |\mathsf{F}\Delta_{\Delta_i^X}\rangle_{\mathsf{F}\Delta_{\Delta_i^X}} \to |\mathsf{F}\Delta_{\Delta_i^X} \oplus \Delta_i^Y\rangle_{\mathsf{F}\Delta_{\Delta_i^X}}$ // Fill FΔ
4 $\quad |\Delta_i^Y\rangle_{D_i^Y} \to |\Delta_i^Y \oplus \mathsf{F}\Delta_{\Delta_i^X}\rangle_{D_i^Y}$ // Empty Δ_i^Y
5 **for** $i = 1, ..., 2^m$ **do**
6 $\quad$ **if** $\mathsf{F}\Delta_i \neq 0$ **then** // For non-zero entries
7 $\quad\quad |\Delta_a^X\rangle_{D_a^X} \to |\Delta_a^X \oplus i\rangle_{D_a^X}$ // Empty Δ_a^X
8 $\quad\quad |a\rangle_A \to |a+1\rangle_A$ // Keep Δ^X sorted properly
9 $|a\rangle_A \to |a\rangle_A$ // Uncompute register A
10 **return** $|\mathsf{F}\Delta'\rangle_{\mathsf{F}\Delta}|\Delta'\rangle_D$ // FΔ' is the filled Fourier
// database, Δ' is the empty compressed database

Dec and allow the size of Δ to be not fixed to q (or the number of non-zero entries in FΔ for Dec^{-1}), which as a result turns Dec into an isometry (and requires to take Dec^* instead of Dec^{-1} for the compressing).

We will show that Equation 2.12 holds.

Theorem 2.3.1. *Let* CFO *be defined as in Algorithm 1 and* Dec *as in Algorithm 2. Then for any query of the form* $|x, \eta\rangle_{XY} \otimes |\Delta\rangle_D$ *it holds that*

$$\mathsf{Dec} \circ \mathsf{CFO}(|x, \eta\rangle_{XY} \otimes |\Delta\rangle_D) = \mathsf{FO} \circ \mathsf{Dec}(|x, \eta\rangle_{XY} \otimes |\Delta\rangle_D). \tag{2.13}$$

Proof. We will begin by writing out the action of CFO on the query,where we will refer to the different subroutines of Algorithm 1 as follows:

- Lines 1-2: Add.

- Line 3: Upd.
- Lines 4-6: Rem.

As CFO starts with Add, which results into adding $(x, 0)$ to Δ in the case that $x \notin \Delta^X$, we can assume there to be a pair $(x, \eta') \in \Delta$. Also by looking at the construction of both CFO and Dec (and specifically the **for** statements), we see that we can simplify to the case of $q = 1$. The only difference with the $q > 1$ case rests in the proper sorting of Δ, for which we refer to Appendix A. CFO now updates the entry containing x by running Upd and afterwards in Rem checks whether this results in an incorrect entry that should be removed.

$$\begin{aligned} |x,\eta\rangle_{XY} \otimes |x,\eta'\rangle_D) &\xrightarrow{\mathsf{Upd}} |x,\eta\rangle_{XY} \otimes |x,\eta' \oplus \eta\rangle_D) \\ &\xrightarrow{\mathsf{Rem}} \begin{cases} |x,\eta\rangle_{XY} \otimes |x,\eta' \oplus \eta\rangle_D), & \text{for } \eta' \neq \eta \\ |x,\eta\rangle_{XY} \otimes |0,0\rangle_D), & \text{for } \eta' = \eta \end{cases}. \end{aligned} \tag{2.14}$$

Now applying Dec on the resulting state will yield

$$\begin{aligned} &\mathsf{Dec} \begin{cases} |x,\eta\rangle_{XY} \otimes |x,\eta' \oplus \eta\rangle_D), & \text{for } \eta' \neq \eta \\ |x,\eta\rangle_{XY} \otimes |0,0\rangle_D), & \text{for } \eta' = \eta \end{cases} \\ &= |x,\eta\rangle_{XY} \otimes |0\rangle_{\mathsf{F}\Delta_1} \otimes \cdots \otimes |\eta' \oplus \eta\rangle_{\mathsf{F}\Delta_x} \otimes \cdots \otimes |0\rangle_{\mathsf{F}\Delta_{2^m}} \\ &= |x,\eta\rangle_{XY} \otimes |\mathsf{P}_{x,(\eta' \oplus \eta)}\rangle_{\mathsf{F}\Delta}. \end{aligned} \tag{2.15}$$

The right side of Equation 2.13 becomes

$$\begin{aligned} \mathsf{FO} \circ \mathsf{Dec}(|x,\eta\rangle_{XY} \otimes |x,\eta'\rangle_D) &= \mathsf{FO}\left(|x,\eta\rangle_{XY} \otimes |0\rangle_{\mathsf{F}\Delta_1} \otimes \cdots \otimes |\eta'\rangle_{\mathsf{F}\Delta_x} \otimes \cdots \otimes |0\rangle_{\mathsf{F}\Delta_{2^m}}\right) \\ &= \mathsf{FO}\left(|x,\eta\rangle_{XY} \otimes |\mathsf{P}_{x,(\eta')}\rangle_{\mathsf{F}\Delta}\right) \\ &= |x,\eta\rangle_{XY} \otimes |\mathsf{P}_{x,\eta'} \oplus \mathsf{P}_{x,\eta}\rangle_{\mathsf{F}\Delta} \\ &= |x,\eta\rangle_{XY} \otimes |\mathsf{P}_{x,(\eta' \oplus \eta)}\rangle_{\mathsf{F}\Delta}. \end{aligned} \tag{2.16}$$

□

Theorem 2.3.1 shows us that CFO is equivalent to FO for the adversary and thus by Lemma 2.2.1 and Lemma 2.2.2 also to StO and PhO. More formally, let $\mathcal{A}^{\mathsf{CFO}}$ be an algorithm interacting with CFO and $\mathcal{A}^{\mathsf{FO}}$ the same algorithm but interacting with FO. Then the equivalence of CFO and FO means that $\Pr[\mathcal{A}^{\mathsf{CFO}} \text{ returns } 1] = \Pr[\mathcal{A}^{\mathsf{FO}} \text{ returns } 1]$. As in most applications the Fourier basis is less intuitive than the standard basis, it would actually be preferred to have a compressed variant of the regular StO and PhO. This is achieved by just applying a basis transformation to CFO. We will call these variants CStO and CPhO.

$$\begin{aligned} \mathsf{CStO} &= \left(\mathbb{I}^X \otimes \mathsf{QFT}^Y \otimes \mathsf{QFT}^D\right) \mathsf{CFO} \left(\mathbb{I}^X \otimes \mathsf{QFT}^Y \otimes \mathsf{QFT}^D\right), \\ \mathsf{CPhO} &= \left(\mathbb{I}^X \otimes \mathbb{I}^Y \otimes \mathsf{QFT}^D\right) \mathsf{CFO} \left(\mathbb{I}^X \otimes \mathbb{I}^Y \otimes \mathsf{QFT}^D\right). \end{aligned} \tag{2.17}$$

2.4. Parallel queries

We can extend all the previously mentioned oracle variations from the sequential query model to the p-parallel query model [JMW17] . Here any adversary interacting with the oracle makes up to p queries at once, which looks as follows for StO:

$$\mathsf{StO}\left(|x_1, y_1, ..., x_p, y_p\rangle_{XY} \otimes |D\rangle_D\right) = |x_1, y_1 \oplus D(x_1), ..., x_p, y_p \oplus D(x_p)\rangle_{XY} \otimes |D\rangle_D. \tag{2.18}$$

From here on we will abbreviate $\boldsymbol{x} := x_1, ..., x_p$. This notation will be extended to all $p-$tuples, for instance $\boldsymbol{x}, \boldsymbol{y} := x_1, y_1, ..., x_p, y_p$.

Just as in the sequential case we can apply QFT on all the Y-registers of the adversary, which results in

$$\mathsf{PhO}\left(|\boldsymbol{x}, \boldsymbol{\eta}\rangle_{XY} \otimes |D\rangle_D\right) = (-1)^{\boldsymbol{\eta}\cdot D(\boldsymbol{x})}|\boldsymbol{x}, \boldsymbol{\eta}\rangle_{XY} \otimes |D\rangle_D. \tag{2.19}$$

Now recall for FO (and thus also CFO) that after every single query only the database register is modified (note that the adversary's register does get entangled with the database register in the case of $\eta \neq 0$). But the database has no way of registering whether the adversary is making p queries in parallel or sequential! Thus in the following chapters when we analyse the state of the database after a p-parallel query to CFO we can simply assume that the adversary is making p sequential queries, with the extra constraint that these p queries are non-adaptive.

From the above we see that when an adversary queries the same $|x, \eta\rangle$ twice in parallel to FO/CFO, this does not affect FΔ/Δ. While this seems unexpected, it is consistent with the definition of PhO for parallel queries. The phase of the output state $(-1)^{\eta\cdot D(x)}$ is normally thought of to be returned to the adversary's register, but we can also choose to transfer it to the database register. Since $(-1)^{\eta\cdot D(x)}(-1)^{\eta\cdot D(x)} = 1$, the output state is unaffected.

2.5. Implementation

Some of the readers might already be convinced that CFO is a unitary operation, as all of mentioned query cases are reversible. Nevertheless we provide a full detailed implementation of CFO in Appendix A, consisting of only unitary operations. This implementation has also been fully programmed in ProjectQ to test whether the detailed algorithm was correct [Zur19]. The number of qubits used by this implementation is $(q+1)(n+m+1) + \max\{n, q+3m\}$. Due to hardware limitations we were only able to simulate the database up to 25 qubits, our most used configuration has been $q = 3, n = 1, m = 2$.

2.5.1. ProjectQ

ProjectQ is a Python module created by [HST17] to allow for the simulation of quantum computers. We will briefly discuss the syntax used in our own program.

Every ProjectQ program contains the following two lines of code:

$$\text{eng = MainEngine()}, \qquad \text{eng.flush()}, \tag{2.20}$$

which respectively create the quantum circuit and run the circuit.

Qubit registers can be added to the circuit by calling

$$\text{eng.allocate\_qubit()}, \qquad \text{eng.allocate\_qureg(n)}, \tag{2.21}$$

for respectively a single qubit register or an n-qubit register. Such a n-qubit register is a list of single qubit registers and inherits all list features in Python. The qubits in these new registers are automatically initialised to the all-zero state.

To increase performance we can delete a register A by using

$$\text{del A}. \tag{2.22}$$

Note that this does require the qubits in this register to be set to the all-zero state before deletion.

As an illustration, assume that we have two registers: A contains a single qubit state and B contains a $n-$qubit state. We can apply the Hadamard gate H as follows to these states:

$$\text{H | A}, \qquad \text{All(H) | B}, \qquad \text{All(H) | (A, B)}. \tag{2.23}$$

The first line of code applies H to register A, the second line applies $H^{\otimes n}$ to register B and the third line applies $H^{\otimes(n+1)}$ to $A \otimes B$. The same syntax works for all other gates and measurements.

Lastly we discuss how to implement controlled gates in ProjectQ. If we want to apply the H gate on each qubit in register B controlled on register A, this can be realised by

$$\begin{aligned} &\text{C(All(H), len(A)) | (A, B)}, \qquad &&\text{with Control(eng, A):} \\ &&&\quad \text{All(H) | B}. \end{aligned} \tag{2.24}$$

The first option is preferred when only a single gate controlled, while the second option allows to control a whole block of code. For all other ProjectQ syntax which we have not implemented we refer to [HST17].

2.5.2. CFO.py

In [Zur19] we have created a program that simulates CFO and allows the user to query this oracle. After starting the program the user is faced (Figure 2.1) with the option of either viewing a random query to a random initialized database for parameters $q = 3, n = 1, m = 2$ or to truly interact with a simulated CFO.

```
Press (0) for a random test query, or (1) for a full simulation
```

Figure 2.1.: Choice to either run a test query or simulate CFO.

```
Running Locate
Finished Locate
Running Add
Starting Permute_inv
Finished Permute_inv
Finished Add
Running Update
Finished Update
Running Remove
Running Permute
Finished Permute
Running Locate
Finished Locate
Running Locate
Finished Locate
Finished Remove
Running Locate
Finished Locate
Running Locate
Finished Locate
```

(a) Progress during a query.

```
Database before the query:
Entry 1: [0, 0] | [1]
Entry 2: [1, 0] | [1]
Entry 3: [0, 0] | [0]
Query: [1, 1] | [1]
Database after the query:
Entry 1: [0, 0] | [1]
Entry 2: [1, 0] | [1]
Entry 3: [1, 1] | [1]
Number of qubits used: 25
```

(b) Joint state before and after the query.

Figure 2.2.: Output during and after a test query.

After choosing option 0, the program starts showing the progress through all subroutines of the total algorithm (Figure 2.2a). After this the program displays the joint state, both from before and after the query, along with the number of qubits used by the algorithm (Figure 2.2b).

If the user chooses option 1, a prompt appears to set the register sizes and input the query (Figure 2.3). We strongly recommend to stay under 25 total qubits, as a higher number might crash the device that the program runs on. Just as in option 0 the program starts showing the progress through all subroutines, before giving the user another choice: stop the program or query again (Figure 2.4). If the user decides to stop the simulation the final state of the database is returned (Figure 2.5).

```
X size: 2
Y size: 1
Database size: 3
Input the X register bitstring: 01
Input the Y register bitstring: 1
```

Figure 2.3.: Specify parameters for the simulation

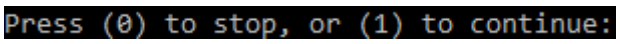

Figure 2.4.: Continue querying or measure the database

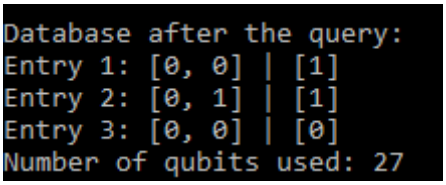

Figure 2.5.: Database after two queries

2.5.3. Challenges

When programming a quantum algorithm, one has to be wary as even the most fundamental programming commands may not so easily translate into a unitary operation. While maintaining an ordered database of pairs could be implemented in a simple fashion classically, our implementation owes most of its complexity due to the fact that sorting is inherently non-reversible. This requires us for every new query where this query should be added to the database or in the case of a removal how to properly resort the database again. While classically one is allowed to change the value of a variable that we condition on during an 'if statement', this is again a non-reversible operation. In the quantum setting this requires extra auxiliary qubits, which at the end of the algorithm need to be recomputed into their original values.

3. Quantum Query Complexity for Non-iterative Relations

3.1. Connection to compressed oracles

In complexity theory there are several problems where an adversary is tasked to find input and output pairs of a given function satisfying some set of constraints. Well known examples of these problems are the Collision problem and the K-sum problem. We will analyze and find lower bounds on the average-case quantum query complexity of these problems by setting the underlying function that the adversary can query to a random function from m to n bits, $F : \{0,1\}^m \to \{0,1\}^n$.

3.1.1. Quantum query solvability

While in complexity theory one is usually interested in a lower bound on the worst-case complexity of the number of quantum queries needed to solve the problem (*quantum query complexity*), in cryptography one is more interested in upper bounding the probability of success in the average case (*quantum query solvability* [Zha15]). The former can also be easily derived from the latter, but not vice versa (which should be reason enough for any mathematician to prefer the latter). Other reasons as to why quantum query solvability is preferred over quantum query complexity are:

- While quantum query complexity is asymptotic in the number of queries, there are problems (such as the quantum oracle interrogation problem [Dam98]) where the probability of success can become trivial by just adding a single query. As a result non-asymptotic bounds in q are preferred.
- In the context of keyed cryptographic systems, the worst case might only prove the security for a certain set of keys, while the average case proves security for randomly chosen keys.
- Quantum query complexity says something about the queries needed for constant success probability, while in cryptography we also want our schemes to be secure for lower-than-constant success probabilities (if the success probability of breaking into a bank would be proportional to $\frac{1}{\log(n)}$, some cryptoanalysts might be tempted to change into a more criminal and lucrative career path).

3.1.2. Relations

In the following chapters we will model our complexity problems in the context of *relations*. Such a relation R^r for $r \geq 0$ on some set X can be viewed as a function $\mathsf{R}^r : X^r \to \{0,1\}$, where an element $x \in X^r$ satisfies R^r whenever $\mathsf{R}^r(x) = 1$. The case $r = 0$ results in the trivial relation, which is always satisfied. For a non-trivial example, we can translate the Collision problem to the relation $\mathsf{R}^r_{\mathsf{Col}}$ as follows:

Definition 3.1.1. *An r-tuple $(x_1, y_1), ..., (x_r, y_r) \in (\{0,1\}^m \times \{0,1\}^n)^r$ satisfies the Collision relation $\mathsf{R}^r_{\mathsf{Col}}$ when:*

- $\forall i, j \in [r]$ *it holds that* $x_i \neq x_j \iff i \neq j$.
- $\forall i, j \in [r]$ *it holds that* $y_i = y_j$.

Thus finding an r-collision is equivalent to finding an r-tuple that satisfies $\mathsf{R}_{\mathsf{Col}}$.

In the context of collisions it might be confusing how many distinct collisions the database precisely contains. As an illustration, let our database consist of the input and output pairs $(0,1), (1,1), (2,1), (3,1)$. One could argue that this database contains $\binom{4}{2} = 6$ distinct 2-collisions. This seems contradictory however, as then there are more collisions than actual database entries. To avoid this confusion, we will precisely define whenever we view two collisions as different, or in general, whenever we view two r-tuples satisfying some relation R of size r are distinct.

Definition 3.1.2. *Two r-tuples $(x_1, y_1), ..., (x_r, y_r), (x'_1, y'_1), ..., (x'_r, y'_r) \in (\{0,1\}^m \times \{0,1\}^n)^r$ are called distinct if $\forall i, j \in [r]$ it holds that $x_i \neq x'_j$.*

Whenever a relation does not satisfy this property we refer to it as a *non-iterative* relation. Returning to our example of the database containing $(0,1), (1,1), (2,1), (3,1)$, we see that the database contains

- 4 distinct collisions of size 1,
- 2 distinct collisions of size 2,
- 1 distinct collision of size 3,
- 1 distinct collision of size 4.

In [Zha18] the connection is shown between the compressed oracle and quantum query bounds of random functions. As seen in Section 2.3, all the information the adversary knows about the random function F can be represented by the entries of the database Δ of CFO. To have a better understanding of the adversary's queries, we apply QFT on the database registers after having done the queries and thus look at D instead of Δ and CPhO instead of CFO. Whenever the adversary is tasked with finding for instance a x such that $F(x) = y$, the adversary can only know this x if the database D contains the pair (x, y). This does ignore the case that the adversary did not know x, but just happened to be lucky enough to guess it. This is formalized in Lemma 5 from [Zha18], which we have slightly rephrased below to be consistent with our notation:

Lemma 3.1.3. *Consider a quantum algorithm A making quantum queries to a random oracle $F : \{0,1\}^m \to \{0,1\}^n$. Let R be some relation of size r and suppose with probability p, A outputs an r-tuple that satisfies R and $H(x_i) = y_i$ for all i. Consider running A with the oracle CPhO and suppose the database D is measured after A produces its output. Let p' be the probability that the r-tuple satisfies R and $(x_i, y_i) \in D$ for all i. Then $\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{r}{N}}$.*

Our analysis consists of finding an upper bound on the probability of the database containing tuples that satisfy some relation R. Then by applying the above lemma we can bound the probability that any quantum algorithm can solve the corresponding complexity problem and then derive a lower bound on the query complexity of the respective problem. In the following chapters we will always assume the total number of queries qp to be larger than k, as otherwise the database will never contain the desired k distinct $r-$tuples.

3.2. Sum problem

As the title of the chapter suggests, we make a distinction between iterative and non-iterative relations.

Definition 3.2.1. *A relation R^r is called* iterative *if $\forall (x_1, y_1), ..., (x_r, y_r)$ satisfying R^r and $\forall t \in [r]$ it holds that $(x_1, y_1), ..., (x_r, y_r)$ contains a $(r-t)$-subtuple satisfying R^{r-t}.*

Going back to our example, we see that $\mathsf{R}^r_{\mathsf{Col}}$ is an iterative relation, since every $(r-t)$-subtuple of an r-collision forms a $(r-t)$-collision. Another example of an iterative relation would be the trivial relation, which is satisfied by every tuple. Upper bounding the probability of success of these iterative relations proves to be slightly harder than the case with non-iterative relations, so to get a better understanding of the technique we start with the non-iterative relations first. Before immediately jumping to a general bound, we start with an example of a non-iterative relation: the K-sum problem. Since we use the variable r to specify the length of relations, we will refer to this problem as the Sum problem to avoid confusing the variables.

Definition 3.2.2. *An r-tuple $(x_1, y_1), ..., (x_r, y_r) \in (\{0,1\}^m \times \{0,1\}^n)^r$ satisfies the Sum relation $\mathsf{R}^r_{\mathsf{Sum}}$ when:*

- *$\forall i, j \in [r]$ it holds that $x_i \neq x_j$ if and only if $i \neq j$.*
- *$\sum_{i=1}^r y_i = 0$.*

Here we view the $y_i \in \mathbb{Z}_N$ under modular addition. Since it does not always hold that there exists an $(r-1)$-sized subset of $\{y_1, ..., y_r\}$ with elements summing up to 0, we see that the Sum problem is a non-iterative relation. In this section we work towards the following theorem:

Theorem 3.2.3. *Let $P_{r,k}$ denote the projector which projects onto the database containing at least k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Sum}}$, as defined in Definition 3.2.2. Denote by $|\psi_q\rangle$ the joint*

state of the adversary and the database after q queries. Then for $k \geq 1$ and constant $r \geq 1$ it holds that

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{q^{r-1}}}{k\sqrt{N}}\right)^k. \tag{3.1}$$

Before we continue, there are some useful identities which we will use in our proofs:

- $\forall k, i \geq 0$ it holds that $\|P_{0,k}|\psi_i\rangle\| = 1$, since the only relation of size 0 is the trivial relation.
- $\forall r, i \geq 0$ it holds that $\|P_{r,0}|\psi_i\rangle\| = 1$, since there are always at least 0 r-tuples satisfying the relation.
- $\forall k, r \geq 1$ it holds that $\|P_{r,k}|\psi_0\rangle\| = 0$, since an empty database can only satisfy the trivial relation.

To prove Theorem 3.2.3, we look at how much the norm $\|P_{r,k}|\psi_i\rangle\|$ increases after every query.

$$\begin{aligned}
\|P_{r,k}|\psi_{i+1}\rangle\| &= \|P_{r,k}\mathsf{CPhO}|\psi_i\rangle\| \\
&= \|P_{r,k}\mathsf{CPhO}P_{r,k}|\psi_i\rangle + P_{r,k}\mathsf{CPhO}(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \\
&\leq \|P_{r,k}\mathsf{CPhO}P_{r,k}|\psi_i\rangle\| + \|P_{r,k}\mathsf{CPhO}(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \\
&= \|P_{r,k}\mathsf{CPhO}|\psi_{i,k_r}\rangle\|\|P_{r,k}|\psi_i\rangle\| + \|P_{r,k}\mathsf{CPhO}|\psi_{i,\overline{k}_r}\rangle\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \\
&\leq \|P_{r,k}|\psi_i\rangle\| + \|P_{r,k}\mathsf{CPhO}|\psi_{i,\overline{k}_r}\rangle\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\|,
\end{aligned} \tag{3.2}$$

where we applied the triangle inequality in the first inequality and in the third equality we introduced the subscripts $|\psi_{i,k_r}\rangle = \frac{P_{r,k}|\psi_i\rangle}{\|P_{r,k}|\psi_i\rangle\|}$, $|\psi_{i,\overline{k}_r}\rangle = \frac{(\mathbb{I}-P_{r,k})|\psi_i\rangle}{\|(\mathbb{I}-P_{r,k})|\psi_i\rangle\|}$.

An important remark is that the projectors in the set $\{P_{r,k} : r, k \geq 0\}$ all commute with each other, since these are defined by projecting onto a set of computational basis states. An explicit construction of $P_{r,k}$ is shown in Appendix B. For there to be $k \geq 1$ distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Sum}}$ after the query, there had to have been at least $k-1$ distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Sum}}$ before the query. Rephrasing this in term of our projectors yields

$$\begin{aligned}
&\|P_{r,k}(\mathsf{CPhO}|\psi_{i,\overline{k}_r}\rangle)\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \\
&= \|P_{r,k}\mathsf{CPhO}P_{r,k-1}|\psi_{i,\overline{k}_r}\rangle + P_{r,k}\mathsf{CPhO}(\mathbb{I} - P_{r,k-1})|\psi_{i,\overline{k}_r}\rangle\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \\
&= \|P_{r,k}\mathsf{CPhO}P_{r,k-1}|\psi_{i,\overline{k}_r}\rangle\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \\
&= \|P_{r,k}\mathsf{CPhO}|\psi_{i,\overline{k}_{r,(k-1)_r}}\rangle\|\|P_{r,k-1}|\psi_{i,\overline{k}_r}\rangle\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \\
&= \|P_{r,k}\mathsf{CPhO}|\psi_{i,\overline{k}_{r,(k-1)_r}}\rangle\|\|P_{r,k-1}(\mathbb{I} - P_{r,k})|\psi_i\rangle\|,
\end{aligned} \tag{3.3}$$

where in the third equality we introduced the subscript $|\psi_{i,\overline{k}_{r,(k-1)_r}}\rangle = \frac{P_{r,k-1}|\psi_{i,\overline{k}_r}\rangle}{\|P_{r,k-1}|\psi_{i,\overline{k}_r}\rangle\|}$. To shorten our notation, we will use an apostrophe to denote the state after a query to CPhO:

$$\|P_{r,k}\mathsf{CPhO}|\psi_{i,\overline{k}_{r,(k-1)_r}}\rangle\| = \|P_{r,k}|\psi'_{i,\overline{k}_{r,(k-1)_r}}\rangle\|. \tag{3.4}$$

We introduce a lemma which will aid us in proving Theorem 3.2.3.

Lemma 3.2.4. *Let $P_{r,k}$ be the projector which projects onto the database containing at least k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Sum}}$, as defined in Definition 3.2.2. Then for $k \geq 1$ and constant $r \geq 1$ it holds that*

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-1)_r}\rangle\| \leq \sqrt{\frac{q^{r-1}}{N}}. \tag{3.5}$$

.

Proof. We start by taking a more detailed look at our state before the query:

$$|\psi_{i,\overline{k}_r,(k-1)_r}\rangle = \sum_{x,\eta,z,D} \alpha_{x,\eta,z,D}|x,\eta,z\rangle_A \otimes |D\rangle_D, \tag{3.6}$$

where D contains all the necessary requirements specified by the subscript of ψ, i.e. D contains precisely $(k-1)$ $r-$tuples satisfying $\mathsf{R}^r_{\mathsf{Sum}}$, as defined in Definition 3.2.2. Let us split the basis states into the following cases:

- $|\text{Zero}\rangle : |x,0,z\rangle_A \otimes |D\rangle_D$.
- $|\text{Add}\rangle : |x,\eta\neq 0,z\rangle_A \otimes |D\rangle_D$ such that $\forall y \in \{0,1\}^n \nexists (x,y) \in D$.
- $|\text{Change}\rangle : |x,\eta\neq 0,z\rangle_A \otimes |D\rangle_D$ such that $\exists (x,y) \in D$ with $y \in \{0,1\}^n$.

We will analyze the state after the query and corresponding norm after projecting with $P_{r.k}$ for each of these cases. Before we do so though let us introduce $G_{\mathsf{R}^r_{\mathsf{Sum}},1}$ as the set of 'good' w's. This set contains the possible values for w such that adding (x,w) for some x to the database completes a $(r-1)$-tuple into an r-tuple satisfying $\mathsf{R}^r_{\mathsf{Sum}}$.

- For any basis state in the case of $|\text{Zero}\rangle$ we have

$$\mathsf{CPhO}\left(|x,0,z\rangle_A \otimes |D\rangle_D\right) = |x,0,z\rangle_A \otimes |D\rangle_D, \tag{3.7}$$

  since CPhO leaves the database invariant whenever $\eta = 0$ After the projection it holds that

$$\|P_{r,k}\mathsf{CPhO}\left(|x,\eta,z\rangle_A \otimes |D\rangle_D\right)\| = 0. \tag{3.8}$$

- For the case of $|\text{Add}\rangle$, we simply add (x,η) to Δ. Returning this to the standard basis results in the following:

$$\begin{aligned}\mathsf{CPhO}\left(|x,\eta,z\rangle_A \otimes |D\rangle_D\right) &\overset{\mathsf{QFT}^D}{=} \mathsf{CFO}\left(|x,\eta,z\rangle_A \otimes |\Delta\rangle_D\right)\\ &= |x,\eta,z\rangle_A \otimes |\Delta \cup (x,\eta)\rangle_D\\ &\overset{\mathsf{QFT}^D}{=} |x,\eta,z\rangle_A \otimes \sum_w \frac{(-1)^{w\cdot\eta}}{\sqrt{N}}|D \cup (x,w)\rangle_D.\end{aligned} \tag{3.9}$$

By using the orthogonality of the basis states we find

$$\begin{aligned}\|P_{r,k}\mathsf{CFO}\left(|x,\eta,z\rangle_A \otimes |D\rangle_D\right)\| &= \sqrt{\|\sum_{w\in G_{\mathsf{R}^r_{\mathsf{Sum}},1}} \frac{(-1)^{w\cdot\eta}}{\sqrt{N}}|D\cup(x,w)\rangle_D\|^2} \\ &= \sqrt{\sum_{w\in G_{\mathsf{R}^r_{\mathsf{Sum}},1}} \|\frac{(-1)^{w\cdot\eta}}{\sqrt{N}}|D\cup(x,w)\rangle_D\|^2} \\ &= \sqrt{\frac{|G_{\mathsf{R}^r_{\mathsf{Sum}},1}|}{N}}. \end{aligned} \tag{3.10}$$

- For the case of $|\text{Change}\rangle$ a slightly larger calculation is necessary. Since $\exists(x,y)\in D$, we can split $D = D'\cup(x,y)$ where $\forall y'\in\{0,1\}^m \nexists(x,y)\in D'$ and accordingly write Δ' for D' in the Fourier basis.

$$\begin{aligned}&\mathsf{CPhO}\left(|x,\eta,z\rangle_A \otimes |D\rangle_D\right) = \mathsf{CPhO}\left(|x,\eta,z\rangle_A \otimes |D'\cup(x,y)\rangle_D\right) \\ &\overset{\mathsf{QFT}^D}{=} \mathsf{CFO}\left(|x,\eta,z\rangle_A \otimes \sum_\lambda \frac{(-1)^{\lambda\cdot y}}{\sqrt{N}}|\Delta'\cup(x,\lambda)\rangle_D\right) \\ &= |x,\eta,z\rangle_A \otimes \left(\sum_{\lambda\neq\eta}\frac{(-1)^{\lambda\cdot y}}{\sqrt{N}}|\Delta'\cup(x,\lambda\oplus\eta)\rangle_D + \frac{(-1)^{\eta\cdot y}}{\sqrt{N}}|\Delta'\rangle_D\right) \\ &\overset{\mathsf{QFT}^D}{=} |x,\eta,z\rangle_A \otimes \left(\sum_{\lambda\neq\eta,w}\frac{(-1)^{\lambda\cdot y}}{\sqrt{N}}\frac{(-1)^{w\cdot(\lambda\oplus\eta)}}{\sqrt{N}}|D'\cup(x,w)\rangle_D + \frac{(-1)^{\eta\cdot y}}{\sqrt{N}}|D'\rangle_D\right),\end{aligned} \tag{3.11}$$

where in the third equality we have distinguished between $\lambda\neq\eta$ and $\lambda=\eta$. By using Equation 1.10 we find

$$\begin{aligned}&|x,\eta,z\rangle_A \otimes \left(\sum_{\lambda\neq\eta,w}\frac{(-1)^{\lambda\cdot y}}{\sqrt{N}}\frac{(-1)^{w\cdot(\lambda\oplus\eta)}}{\sqrt{N}}|D'\cup(x,w)\rangle_D + \frac{(-1)^{\eta\cdot y}}{\sqrt{N}}|D'\rangle_D\right) \\ &= |x,\eta,z\rangle_A \otimes \left(\sum_{\lambda\neq\eta,w}\frac{(-1)^{\lambda\cdot(y\oplus w)}}{\sqrt{N}}\frac{(-1)^{w\cdot\eta}}{\sqrt{N}}|D'\cup(x,w)\rangle_D + \frac{(-1)^{\eta\cdot y}}{\sqrt{N}}|D'\rangle_D\right) \\ &= |x,\eta,z\rangle_A \otimes \left(\sum_{\lambda,w}\frac{(-1)^{\lambda\cdot(y\oplus w)}}{\sqrt{N}}\frac{(-1)^{w\cdot\eta}}{\sqrt{N}}|D'\cup(x,w)\rangle_D\right. \\ &\quad \left. -\sum_w\frac{(-1)^{w\cdot\eta}}{N}|D'\cup(x,w)\rangle_D + \frac{(-1)^{\eta\cdot y}}{\sqrt{N}}|D'\rangle_D\right) \\ &= |x,\eta,z\rangle_A \otimes \left((-1)^{y\cdot\eta}|D\rangle_D - \sum_w\frac{(-1)^{w\cdot\eta}}{N}|D'\cup(x,w)\rangle_D + \frac{(-1)^{\eta\cdot y}}{\sqrt{N}}|D'\rangle_D\right).\end{aligned} \tag{3.12}$$

After the projection it holds that

$$\begin{aligned}\|P_{r,k}\mathsf{CFO}\left(|x,\eta,z\rangle_A \otimes |D\rangle_D\right)\| &= \sqrt{\|\sum_{w\in G_{\mathsf{R}^r_{\mathsf{Sum}},1}} \frac{(-1)^{w\cdot\eta}}{N}|D'\cup(x,w)\rangle_D\|^2} \\ &= \sqrt{\sum_{w\in G_{\mathsf{R}^r_{\mathsf{Sum}},1}} \|\frac{(-1)^{w\cdot\eta}}{N}|D'\cup(x,w)\rangle_D\|^2} \\ &= \sqrt{\frac{|G_{\mathsf{R}^r_{\mathsf{Sum}},1}|}{N^2}}. \end{aligned} \tag{3.13}$$

For the total state we conclude

$$\begin{aligned}\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-1)_r}\rangle\| &= \sqrt{\|P_{r,k}\sum_{x,\eta,z,D}\alpha_{x,\eta,z,D}|x,\eta,z\rangle_A\otimes|D\rangle_D\|^2} \\ &= \sqrt{\|\alpha_{\text{Zero}}P_{r,k}|\text{Zero}\rangle + \alpha_{\text{Add}}P_{r,k}|\text{Add}\rangle + \alpha_{\text{Change}}P_{r,k}|\text{Change}\rangle\|^2} \\ &= \sqrt{\|\alpha_{\text{Zero}}P_{r,k}|\text{Zero}\rangle\|^2 + \|\alpha_{\text{Add}}P_{r,k}|\text{Add}\rangle\|^2 + \|\alpha_{\text{Change}}P_{r,k}|\text{Change}\rangle\|^2} \\ &= \sqrt{\|\alpha_{\text{Add}}\|^2\frac{|G_{\mathsf{R}^r_{\mathsf{Sum}},1}|}{N} + \|\alpha_{\text{Change}}\|^2\frac{|G_{\mathsf{R}^r_{\mathsf{Sum}},1}|}{N^2}} \\ &\leq \sqrt{\frac{|G_{\mathsf{R}^r_{\mathsf{Sum}},1}|}{N}}. \end{aligned} \tag{3.14}$$

We are left with computing the value $|G_{\mathsf{R}^r_{\mathsf{Sum}},1}|$. Since for every $(r-1)$-tuple there exists at most a unique w such that adding (x,w) creates an r-tuple satisfying $\mathsf{R}^r_{\mathsf{Sum}}$, we can bound $|G_{\mathsf{R}^r_{\mathsf{Sum}},1}|$ by the number of $(r-1)$-tuples in the database, which is at most $\binom{q}{r-1} \leq q^{r-1}$ after $i < q$ queries. □

We now possess all the tools to prove Theorem 3.2.3, which we have restated below.

Theorem 3.2.3. *Let $P_{r,k}$ denote the projector which projects onto the database containing at least k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Sum}}$, as defined in Definition 3.2.2. Denote by $|\psi_q\rangle$ the joint state of the adversary and the database after q queries. Then for $k \geq 1$ and constant $r \geq 1$ it holds that*

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{q^{r-1}}}{k\sqrt{N}}\right)^k. \tag{3.15}$$

Proof. By combining Equation 3.2 with Lemma 3.2.4 we arrive at the following recursion:

$$\begin{aligned}
\|P_{r,k}|\psi_q\rangle\| &\leq \sum_{i=0}^{q-1} \sqrt{\frac{q^{r-1}}{N}} \|P_{r,k-1}(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \\
&\leq \sum_{i=0}^{q-1} \sqrt{\frac{q^{r-1}}{N}} \|P_{r,k-1}|\psi_i\rangle\| \\
&\leq \sum_{i_1=0}^{q-1} \sqrt{\frac{q^{r-1}}{N}} \sum_{i_2=0}^{i_1-1} \sqrt{\frac{q^{r-1}}{N}} \|P_{r,k-2}|\psi_{i_2}\rangle\| \\
&\leq \sum_{0\leq i_k<\cdots<i_2<i_1\leq q-1} \left(\sqrt{\frac{q^{r-1}}{N}}\right)^k . \qquad (3.16)
\end{aligned}$$

To be able to iterate all the way to $P_{r,0}$ in the final step we need $q \geq k$, but note that any R^r with $r \geq 1$ is trivial for $q < k$ (because in that case $\|P_{r,k}|\psi_q\rangle\| = 0$). Since $\left(\sqrt{\frac{q^{r-1}}{N}}\right)^k$ is independent of $i_1, ..., i_k$, we can compute $\sum_{0\leq i_k<\cdots<i_2<i_1\leq q-1} = \binom{q}{k}$ separately. Using the inequality that $k! \geq (k/e)^k$, we bound $\binom{q}{k} \leq \frac{q^k}{k!} \leq \left(\frac{eq}{k}\right)^k$ which completes the theorem. □

As we have said at the beginning of this chapter, Lemma 3.1.3 allows us to connect the above theorem to the upper bound on the probability of success.

Corollary 3.2.5. *Any quantum algorithm A making q queries to a random oracle $F : \{0,1\}^m \to \{0,1\}^n$ finds k distinct solutions to the r-Sum problem, for $k \leq \sqrt{\frac{N}{r}}$ and constant $r \geq 1$, with probability at most*

$$\left(\left(\frac{eq\sqrt{q^{r-1}}}{k\sqrt{N}}\right)^k + k\sqrt{\frac{r}{N}}\right)^2 \qquad (3.17)$$

. As a result, any quantum algorithm A needs to make $\Omega\left(k^{2/(r+1)}N^{1/(r+1)}\right)$ queries to find k distinct solutions to the r-Sum problem with constant probability.

If we substitute $k = 1$ our resulting query complexity is consistent with the tight result of [BS12], where they give an query complexity bound of $\Omega\left(M^{r/(r+1)}\right)$ under the assumption that $M \geq N^r$.

3.3. General non-iterative relations

We can adapt Theorem 3.2.3 to the case of any non-iterative relation R. Recall the set of constraints of $\mathsf{R}^r_{\mathsf{Sum}}$:

- $\forall i, j \in [r]$ it holds that $x_i \neq x_j \iff i \neq j$.
- $\sum_{i=1}^r y_i = 0$.

We never took the first r constraints into account when proving the upper bound, as these are automatically satisfied by the construction of our database. General relations could however have different constraints on the x_i values. We want to find a way to remove these constraints, such that when adding a new (x, y) value to the database after a new query, we only have to check the y-value if we want to know whether this new entry has created a new r-tuple satisfying R^r. This will allow us to keep our proof analogous to the $\mathsf{R}^r_{\mathsf{Sum}}$ variant.

We introduce the following procedure which reduces any relation R^r to a maximally reduced relation $\hat{\mathsf{R}}^{\hat{r}}$.

- Always remove every constraint on x_i which only depends on all other x_j's.
- If for every constraint on x_i there exists a value of y_i such that the constraint is not satisfied, we call the relation maximally reduced.
- Otherwise, remove this constraint and replace x_i by its constraint in every other constraint in the relation.

The reasoning behind these steps is that without loss of generality we can change the order of the queries such that all other x_j, y_j's have already been queried by the adversary before x_i is queried. The adversary can measure these registers on the adversary's side such that when he queries a new x_i this x_i will satisfy the needed constraint. There is no method for the adversary however to have any information about y_i before querying x_i, because we are querying random functions. As a result such a constraint can not be removed.

To give an example of a reduction, let us look at the following set of constraints for some $r \geq 3$:

- $\forall i \in \{2, ..., r-1\}$ it holds that $x_{i+1} = y_i$.
- $\sum_{i=1}^{r} x_i = 0$.
- $x_1 = x_r + y_1$.

Note that by the structure of our database it will always hold that $\forall i, j \in [r]\ x_i \neq x_j \iff i \neq j$, even if this is not a constraint. We can immediately remove the $(r-1)$-th constraint (the second item in the list above) as it is not dependent on any of the y_j.

- $\forall i \in \{2, ..., r-1\}$ it holds that $x_{i+1} = y_i$.
- $x_1 = x_r + y_1$.

For the first $r-2$ constraints, every x_{i+1} is independent of y_{i+1} and thus there exists no value of y_{i+1} such that the constraint is not satisfied. As a result, we remove these constraints and replace x_{i+1} by y_i in all other constraints:

- $x_1 = y_{r-1} + y_1$.

For the final constraint, there exists a value of y_1 for every value of y_{r-1} such that the constraint does not hold. This means that there is no constraint left that can be removed and we conclude that the relation is maximally reduced. We also note that there is no longer any constraint on the values of $(x_2, y_2), ..., (x_{r-2}, y_{r-2})$, so this new relation actually only size 2.

A reduction does require q to be larger than the number of removed constraints, but as this number is bounded by r this merely makes our bound less tight in the case that $q < r$. Since r is constant our final bound remains unaffected in the asymptotic case.

We conclude the main result for non-iterative relations:

Theorem 3.3.1. *Let $P_{r,k}$ denote the projector which projects onto the database containing at least k distinct r-tuples satisfying the maximally reduced and non-iterative relation R^r. Denote by $|\psi_q\rangle$ the joint state of the adversary and the database after q queries. Then for $k \geq 1$ and constant $r \geq 1$ it holds that*

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{|G_{\mathsf{R}^r,1}|}}{k\sqrt{N}}\right)^k. \tag{3.18}$$

Proof. As R^r is maximally reduced, we can simply repeat the proofs of Lemma 3.2.4 and Theorem 3.2.3, where we omit the calculation of $|G_{\mathsf{R}^r,1}|$. □

3.4. Parallel queries for non-iterative relations

When the adversary is allowed to make p-parallel queries to the oracle, our analysis from the previous section is no longer correct. More specifically, we have stated that for there to be $k \geq 1$ distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Sum}}$ after the query, there had to have been at least $k-1$ distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Sum}}$ before the query. In the parallel case however, we can create up to p distinct r-tuples satisfying any R^r within a single p-parallel query. '

By applying the triangle inequality the parallel analog of Equation 3.3 becomes

$$\|P_{r,k}(\mathsf{CFO}|\psi_{i,\overline{k}_r}\rangle)\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \leq \sum_{j=1}^{\min\{p,k\}} \|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r}\rangle\|\|P_{r,k-j}(\mathbb{I} - P_{r,k})|\psi_i\rangle\|. \tag{3.19}$$

Lemma 3.2.4 also changes, as we are no longer restricted to completing any $(r-1)$-tuple, but we can go as low as any $(\max\{0, r-p\})$-tuple. We extend our notation of $G_{R,1}$ from the sequential case as follows: denote by $G_{\mathsf{R}^r,t}$ the set of possible values for $w_1, ..., w_t$ such that adding $(x_1, w_1), ..., (x_1, w_t)$ for some $x_1, ..., x_t$ completes a $(r-t)$-tuple into an r-tuple satisfying R (in the sequential case we only had $t = 1$). We give a generalisation of Lemma 3.2.4 which is also applicable to parallel queries:

Lemma 3.4.1. *Let $P_{r,k}$ be the projector which projects onto the database containing at least k distinct r-tuples satisfying the maximally reduced relation R^r. Then for $k \geq 1, j \leq [\min\{p,r\}]$, constant $r \geq 1$ and parallel query size p it holds that*

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r}\rangle\| \leq \left(\min\{p,r\}\sqrt{\frac{|G_{\mathsf{R}^r,t_{\max}}|(ep)^{t_{\max}}}{(t_{\max}N)^{t_{\max}}}}\right)^j, \tag{3.20}$$

where $t_{\max} := \underset{t\leq\lfloor\min\{p,r\}/j\rfloor}{\arg\max}\left\{\frac{|G_{\mathsf{R}^r,t}|(ep)^t}{(tN)^t}\right\}$.

Proof. We again start by taking a more detailed look at our state before the query, where we have denoted $(x_1,\eta_1),...,(x_p,\eta_p)$ by $(\boldsymbol{x},\boldsymbol{\eta})$:

$$|\psi_{i,\overline{k}_r,(k-j)_r}\rangle = \sum_{\boldsymbol{x},\boldsymbol{\eta},z,D,w_1,...,w_p} \alpha_{\boldsymbol{x},\boldsymbol{\eta},z,D}|\boldsymbol{x},\boldsymbol{\eta},z\rangle_A \otimes |D\rangle_D, \tag{3.21}$$

where D contains all the necessary requirements specified by the subscript of ψ. Recall from the proof of Lemma 3.2.4 that for a single query the basis states in $|\text{Add}\rangle$ increase the norm of the projection the most. Since it does not affect the database register whether the queries $(\boldsymbol{x},\boldsymbol{\eta})$ are queried sequentially or in parallel, we can bound the norm after the projection by the case that each $|x,\eta\neq 0,z\rangle_A \otimes |D\rangle_D \in |\text{Add}\rangle$. The state after the query of these resulting basis states is

$$\mathsf{CFO}\left(|\boldsymbol{x},\boldsymbol{\eta},z\rangle_A \otimes |D\rangle_D\right) = |\boldsymbol{x},\boldsymbol{\eta},z\rangle_A \otimes \sum_{\boldsymbol{w}} \frac{(-1)^{\boldsymbol{w}\cdot\boldsymbol{\eta}}}{\sqrt{N^p}}|D\cup(\boldsymbol{x},\boldsymbol{w})\rangle_D, \tag{3.22}$$

where we abbreviated $\boldsymbol{w} = w_1,...,w_p$.
We write $\boldsymbol{w} \in G_{\mathsf{R}^r,t}$ if $\{\boldsymbol{w}\}$ contains a subset of size t taking on values in $G_{\mathsf{R}^r,t}$. We distinguish the completion of $(r-t)$-tuples for different t and note that $tj \leq p$ and $t \leq \min\{p,r\}$. Combining this yields $t \leq \lfloor \min\{p,r\}/j\rfloor$. With the help of the triangle inequality applying $P_{r,k}$ results in

$$\begin{aligned}
\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r}\rangle\| &\leq \||\boldsymbol{x},\boldsymbol{\eta},z\rangle_A \otimes \sum_{t_1,...,t_j=1}^{\lfloor\min\{p,r\}/j\rfloor} \sum_{\substack{\boldsymbol{w}_1\in G_{\mathsf{R}^r,t_1}\\ \vdots \\ \boldsymbol{w}_j\in G_{\mathsf{R}^r,t_j}}} \frac{(-1)^{\boldsymbol{w}\cdot\boldsymbol{\eta}}}{\sqrt{N^p}}|D\cup(\boldsymbol{x},\boldsymbol{w})\rangle_D\| \\
&\leq \sum_{t_1,...,t_j=1}^{\lfloor\min\{p,r\}/j\rfloor}\left(\sqrt{\frac{|G_{\mathsf{R}^r,t_1}|\binom{p}{t_1}}{N^{t_1}}}\cdots\sqrt{\frac{|G_{\mathsf{R}^r,t_j}|\binom{p}{t_j}}{N^{t_j}}}\right) \\
&\leq \sum_{t_1,...,t_j=1}^{\lfloor\min\{p,r\}/j\rfloor}\left(\sqrt{\frac{|G_{\mathsf{R}^r,t_1}|(ep)^{t_{\max}}}{(t_1N)^{t_1}}}\cdots\sqrt{\frac{|G_{\mathsf{R}^r,t_j}|(ep)^{t_j}}{(t_jN)^{t_j}}}\right) \\
&\leq \left(\lfloor\min\{p,r\}/j\rfloor\sqrt{\frac{|G_{\mathsf{R}^r,t_{\max}}|(ep)^{t_{\max}}}{(t_{\max}N)^{t_{\max}}}}\right)^j.
\end{aligned} \tag{3.23}$$

□

Before we head to the general version of Theorem 3.2.3, we introduce a helpful lemma which aids in proving both the next theorem, as well as the main result in the next chapter. The proof of this lemma can be found in Appendix C

Lemma 3.4.2. *Let the series $a_{q,n}, b_{q,n}$ be defined as follows:*

- $a_{q,0} := q$,
- $b_{q,0} := q$,
- $a_{q,n} := \sum_{i=1}^{n} \sum_{j=0}^{q-1} c^i a_{j,n-i}$,
- $b_{q,n} := \sum_{i=1}^{n} \sum_{j=0}^{q-1} (c^i b_{j,n-i} + d)$,

for constants c, d. Then $\forall n \geq 1$ it holds that

$$\begin{aligned} a_{q,n} &= \frac{q^n}{n!} 2^{n-1} c^n, \\ b_{q,n} &= a_{q,n} + \sum_{i=0}^{n-1} (n-i) a_{q,i} d. \end{aligned} \tag{3.24}$$

Theorem 3.4.3. *Let $P_{r,k}$ denote the projector which projects onto the database containing at least k distinct r-tuples satisfying the maximally reduced and non-iterative relation R^r. Denote by $|\psi_q\rangle$ the joint state of the adversary and the database after q queries. Then for $k \geq 1$, constant $r \geq 1$ and parallel query size p it holds that*

$$\| P_{r,k} |\psi_q\rangle \| \leq \left(\frac{2erq \sqrt{|G_{\mathsf{R}^r, t_{\max}}| (ep)^{t_{\max}}}}{k \sqrt{(t_{\max} N)^{t_{\max}}}} \right)^k. \tag{3.25}$$

Proof. By combining Equation 3.19 with Lemma 3.4.1 we arrive at the following recursion:

$$\begin{aligned} \| P_{r,k} |\psi_q\rangle \| &\leq \sum_{i=0}^{q-1} \sum_{j=1}^{\min\{p,k\}} \left(\min\{p, r\} \sqrt{\frac{|G_{\mathsf{R}^r, t_{\max}}| (ep)^{t_{\max}}}{(t_{\max} N)^{t_{\max}}}} \right)^j \| P_{r,k-j} (\mathbb{I} - P_{r,k}) |\psi_i\rangle \| \\ &\leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} \left(r \sqrt{\frac{|G_{\mathsf{R}^r, t_{\max}}| (ep)^{t_{\max}}}{(t_{\max} N)^{t_{\max}}}} \right)^j \| P_{r,k-j} |\psi_i\rangle \|. \end{aligned} \tag{3.26}$$

Substituting $c = r\sqrt{\frac{|G_{\mathsf{R}^r,t_{\max}}|(ep)^{t_{\max}}}{(t_{\max}N)^{t_{\max}}}}$ and $a_{q,k} = \|P_{r,k}|\psi_q\rangle\|$ into Lemma 3.4.2 yields

$$\begin{aligned}
\|P_{r,k}|\psi_q\rangle\| &\leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\left(r\sqrt{\frac{|G_{\mathsf{R}^r,t_{\max}}|(ep)^{t_{\max}}}{(t_{\max}N)^{t_{\max}}}}\right)^j \|P_{r,k-j}|\psi_i\rangle\| \\
&= \frac{q^k}{k!}2^{k-1}c^j \\
&= \frac{q^k}{k!}2^{k-1}\left(r\sqrt{\frac{|G_{\mathsf{R}^r,t_{\max}}|(ep)^{t_{\max}}}{(t_{\max}N)^{t_{\max}}}}\right)^k \\
&\leq \left(\frac{eq}{k}\right)^k 2^{k-1}\left(r\sqrt{\frac{|G_{\mathsf{R}^r,t_{\max}}|(ep)^{t_{\max}}}{(t_{\max}N)^{t_{\max}}}}\right)^k \\
&\leq \left(\frac{2erq\sqrt{|G_{\mathsf{R}^r,t_{\max}}|(ep)^{t_{\max}}}}{k\sqrt{(t_{\max}N)^{t_{\max}}}}\right)^k . \qquad (3.27)
\end{aligned}$$

□

We can apply this Theorem to $\mathsf{R}^r_{\mathsf{Sum}}$ to bound the quantum query solvability of the Sum problem for p-parallel queries. To do this, we have to derive the value of $|G_{\mathsf{R}^r_{\mathsf{Sum}},t}|$. There are at most $\binom{(q-1)p}{r-t} \leq (qp)^t$ $(r-t)$-tuples in the database after $q-1$ queries. For each such $(r-t)$-tuple, there exist at most N^{t-1} values we can assign to $w_1, ..., w_t$ such that it gets completed into an r-tuple satisfying $\mathsf{R}^r_{\mathsf{Sum}}$. So $|G_{\mathsf{R}^r_{\mathsf{Sum}},t}| \leq (qp)^{r-t}N^{t-1}$ and thus $\frac{|G_{\mathsf{R}^r,t}|(ep)^t}{(tN)^t} \leq \frac{e^t q^{r-t}p^r}{t^t N}$, which is decreasing in t. We conclude $t_{\max} = 1$ and thus

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{2erq\sqrt{eq^{r-1}p^r}}{k\sqrt{N}}\right)^k . \qquad (3.28)$$

Combining this with Lemma 3.1.3 yields

Corollary 3.4.4. *Any quantum algorithm A making q p-parallel queries to a random oracle $F : \{0,1\}^m \to \{0,1\}^n$ finds k distinct solutions to the r-Sum problem, for $k \leq \sqrt{\frac{N}{r}}$ and constant $r \geq 1$, with probability at most*

$$\left(\left(\frac{2erq\sqrt{eq^{r-1}p^r}}{k\sqrt{N}}\right)^k + k\sqrt{\frac{r}{N}}\right)^2 . \qquad (3.29)$$

As a result, any quantum algorithm A needs to make $\Omega\left(k^{2/(r+1)}p^{-r/(r+1)}N^{1/(r+1)}\right)$ p-parallel queries to find k distinct solutions to the r-Sum problem with constant probability.

If we substitute $k = 1$ our resulting query complexity is consistent with the tight result of [JMW17], where they give an query complexity bound of $\Omega\left((M/p)^{r/(r+1)}\right)$ under the assumption that $M \geq N^r$.

3.5. Fully parallel queries for non-iterative relations

In all the previous sections we have taken the liberty to bound $q-1$ by q. In general the effect of this simplification is negligible on our results and does not affect the asymptotic cases. There is however one instance where this bound does affect the final bound, namely whenever $q=1$ i.e. the fully parallel query. To illustrate this, we return to the last step in Equation 3.23.

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r}\rangle\| \leq \sum_{t_1,\ldots,t_j=1}^{\min\{p,r\}} \left(\sqrt{\frac{|G_{\mathsf{R}^r,t_1}|(ep)^{t_{\max}}}{(t_1N)^{t_1}}}\cdots\sqrt{\frac{|G_{\mathsf{R}^r,t_j}|(ep)^{t_j}}{(t_jN)^{t_j}}}\right). \tag{3.30}$$

As in the fully parallel case $q-1=0$, so the only tuples in the database after 0 queries are those of length $0=r-r$. Thus $|G_{\mathsf{R}^r,t}|$ is 0 for any $t\neq r$ and as a result we have

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r}\rangle\| \leq \left(\sqrt{\frac{|G_{\mathsf{R}^r,r}|(ep)^r}{(rN)^r}}\right)^j. \tag{3.31}$$

In the case of $q=1$, the only value for i will be 0. As a result $\|P_{r,k-j}|\psi_i\rangle\| = \delta_{j,k}$. Combining these two results yields

$$\begin{aligned}\|P_{r,k}|\psi_q\rangle\| &\leq \sum_{j=1}^{k}\left(\sqrt{\frac{|G_{\mathsf{R}^r,r}|(ep)^r}{(rN)^r}}\right)^j \|P_{r,k-j}|\psi_i\rangle\| \\ &= \left(\sqrt{\frac{|G_{\mathsf{R}^r,r}|(ep)^r}{(rN)^r}}\right)^k.\end{aligned} \tag{3.32}$$

Corollary 3.5.1. *Any quantum algorithm A making a single fully p-parallel query to a random oracle $F:\{0,1\}^m\to\{0,1\}^n$ finds k distinct solutions to the r-Sum problem, for $k\leq\sqrt{\frac{N}{r}}$ and constant $r\geq 1$, with probability at most*

$$\left(\left(\sqrt{\frac{(ep)^r}{r^rN}}\right)^k + k\sqrt{\frac{r}{N}}\right)^2. \tag{3.33}$$

As a result, any quantum algorithm A needs to make a parallel query of size at least $\Omega\left(N^{1/r}\right)$ to find k distinct solutions to the r-Sum problem with constant probability.

This is precisely the classical bound, where we need $\Omega(M)$ queries under the assumption that $M\geq N^r$.

Another problem in the fully parallel case is the reduction of relations. We can no longer remove a constraint on x_i, as there have not been any $x_1, y_1, ..., x_r, y_r$ queried yet. We can however rewrite this constraint to a constraint on the y values. An example of this will be given in Section 5.1.

4. Quantum Query Complexity for Iterative Relations

4.1. Collision problem

Now that we have analysed the non-iterative relations, it is time for their iterative counterpart. The main idea is that, compared to non-iterative relations, $|G_{\mathsf{R}^r,t_{\max}}|$ is dependent on $\|P_{r-t,k}|\psi_q\rangle\|$. The reason for this being that any $(r-t)$-tuple that can potentially be completed has to satisfy R^{r-t} by the definition of an iterative relation. As a result just applying Theorem 3.4.3 will yield a non-optimal upper bound. As with the non-iterative relations, we start with a specific example to sketch the technique. As such, we will work towards the following theorem in this section:

Theorem 4.1.1. *Let $P_{r,k}$ denote the projector which projects onto the database containing at least k distinct sets of r-tuples satisfying $\mathsf{R}^r_{\mathsf{Col}}$, as defined in Definition 3.1.1. Denote by $|\psi_q\rangle$ the joint state of the adversary and the database after q queries. Also define $\gamma(r) = 2^{(2^{r-2}-1)/2^{r-2}} e^{(2^{r-1}-1)/2^{r-2}}$. Then for $k \leq N$ and constant $r \geq 2$ it holds that*

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{\gamma(r)q^{(2^r-1)/2^{r-1}}}{kN^{(2^{r-1}-1)/2^{r-1}}} + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/(2^{r-1}-1)2^{r+1}}\right)\right)^k + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r}N^{1/2^r}}\right). \tag{4.1}$$

Note that compared to Theorem 3.2.3 we set $r \geq 2$ instead of $r \geq 1$. This results from the fact that $\mathsf{R}^1_{\mathsf{Col}}$ is trivial (whereas most other relations are only trivial for $r = 0$). We follow a similar reasoning as [LZ18], but manage to generalise their result to hold for $k \leq \sqrt{\frac{N}{r}}$ instead of $k \leq N^{1/2^r}$.

The method of proving Theorem 4.1.1 starts similar to the non-iterative case, so we briefly recall Equations 3.2 and 3.3.

$$\|P_{r,k}|\psi_{i+1}\rangle\| \leq \|P_{r,k}|\psi_i\rangle\| + \|P_{r,k}\mathsf{CFO}|\psi_{i,\overline{k}_r}\rangle\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\|. \tag{4.2}$$

$$\|P_{r,k}(\mathsf{CFO}|\psi_{i,\overline{k}_r}\rangle)\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\| = \|P_{r,k}|\psi'_{i,\overline{k}_{r,(k-1)_r}}\rangle\|\|P_{r,k-1}(\mathbb{I} - P_{r,k})|\psi_i\rangle\|. \tag{4.3}$$

The first real change occurs when we look at Lemma 3.2.4. Its iterative counterpart is stated as follows:

Lemma 4.1.2. *Let $P_{r,k}$ and $\hat{P}_{r,k}$ be the projectors which project on the database containing respectively at least and exactly k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Col}}$, as defined in Definition 3.1.1. Then for $k \geq 1$ and constant $r \geq 2$ it holds that*

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-1)_r}\rangle\| \leq \sqrt{\sum_{l=0}^{N} \frac{l}{N} \|\hat{P}_{r-1,l}|\psi_{i,\overline{k}_r,(k-1)_r}\rangle\|^2}. \tag{4.4}$$

Proof. We start with abbreviating $|\psi_{i,\overline{k}_r,(k-1)_r\hat{l}_{r-1}}\rangle = \frac{\hat{P}_{r-1,l}|\psi_{i,\overline{k}_r,(k-1)_r}\rangle}{\|\hat{P}_{r-1,l}|\psi_{i,\overline{k}_r,(k-1)_r}\rangle\|}$ to derive

$$\begin{aligned}\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-1)_r}\rangle\|^2 &= \|P_{r,k}\sum_{l=0}^{N}\hat{P}_{r-1,l}|\psi_{i,\overline{k}_r,(k-1)_r}\rangle\|^2 \\ &= \sum_{l=0}^{N}\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-1)_r,\hat{l}_{r-1}}\rangle\|^2\|\hat{P}_{r-1,l}|\psi_{i,\overline{k}_r,(k-1)_r}\rangle\|^2. \end{aligned}\tag{4.5}$$

Here we used the orthogonality and commutativity of the $\hat{P}_{r-1,l}$ in the second equality. These projectors namely also still commute, as $\hat{P}_{r,k} = (\mathbb{I} - P_{r,k+1})P_{r,k}$. Just as in the non-iterative case (see the proof of Equation 3.2.4) we can bound

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-1)_r,\hat{l}_{r-1}}\rangle\| \leq \sqrt{\frac{|G_{\mathsf{R}^r_{\mathsf{Col}},1,l}|}{N}}. \tag{4.6}$$

Note that we added a l to $G_{\mathsf{R}^r_{\mathsf{Col}},1,l}$ as $\mathsf{R}^r_{\mathsf{Col}}$ is an iterative relation and thus every possible $(r-1)$-tuple that we can complete must satisfy $\mathsf{R}^{r-1}_{\mathsf{Col}}$. In the specific case of collisions, there exists a unique w for every $(r-1)$-collision such that (x,w) completes it into an r-collision and thus we have $G_{\mathsf{R}^r_{\mathsf{Col}},1,l} = l$.

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-1)_r}\rangle\| \leq \sqrt{\sum_{l=0}^{N} \frac{l}{N} \|\hat{P}_{r-1,l}|\psi_{i,\overline{k}_r,(k-1)_r}\rangle\|^2}. \tag{4.7}$$

□

Before we can prove Theorem 4.1.1, we will use the above lemma to prove one final intermediate step.

Lemma 4.1.3. *Let $P_{r,k}$ be the projector which projects onto the database containing at least k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Col}}$, as defined in Definition 3.1.1. Then for any $M_{r-1} \leq N$, $k \geq 1$ and constant $r \geq 2$ it holds that*

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{M_{r-1}}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{M_{r-1}}{N}}}q\|P_{r-1,M_{r-1}+1}|\psi_q\rangle\|. \tag{4.8}$$

Proof. By combining Lemma 4.1.2 with Equations 3.2 and 3.3, we arrive at the following recursion:

$$\begin{aligned}\|P_{r,k}|\psi_q\rangle\| &\leq \sum_{i=0}^{q-1} \sqrt{\sum_{l=0}^{N} \frac{l}{N} \|\hat{P}_{r-1,l}|\psi_{i,\overline{k}_r,(k-1)_r}\rangle\|^2} \cdot \|P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\| \\ &= \sum_{i=0}^{q-1} \sqrt{\sum_{l=0}^{N} \frac{l}{N} \|\hat{P}_{r-1,l} P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\|^2}. \end{aligned} \tag{4.9}$$

We introduce a cut-off point $M_{r-1} \leq N$ for the sum over l (the value of M_{r-1} will be determined later):

$$\begin{aligned}&\sum_{i=0}^{q-1} \sqrt{\sum_{l=0}^{N} \frac{l}{N} \|\hat{P}_{r-1,l} P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\|^2} \\ &= \sum_{i=0}^{q-1} \sqrt{\sum_{l=0}^{M_{r-1}} \frac{l}{N} \|\hat{P}_{r-1,l} P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\|^2 + \sum_{l=M_{r-1}+1}^{N} \frac{l}{N} \|\hat{P}_{r-1,l} P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\|^2} \\ &\leq \sum_{i=0}^{q-1} \sqrt{\frac{M_{r-1}}{N} \sum_{l=0}^{M_{r-1}} \|\hat{P}_{r-1,l} P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\|^2 + \sum_{l=M_{r-1}+1}^{N} \|\hat{P}_{r-1,l} P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\|^2} \\ &\leq \sum_{i=0}^{q-1} \sqrt{\frac{M_{r-1}}{N} \|P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\|^2 + \|P_{r-1,M_{r-1}+1} P_{r,k-1}(\mathbb{1} - P_{r,k})|\psi_i\rangle\|^2} \\ &\leq \sum_{i=0}^{q-1} \sqrt{\frac{M_{r-1}}{N} \|P_{r,k-1}|\psi_i\rangle\|^2 + \|P_{r-1,M_{r-1}+1}|\psi_i\rangle\|^2}, \end{aligned} \tag{4.10}$$

where in the first inequality we have bounded l by the largest value that it takes on in each respective sum. In the second inequality we have used the (in)equalities $\sum_{l=0}^{M_{r-1}} \hat{P}_{r-1,l} \leq \sum_{l=0}^{N} \hat{P}_{r-1,l} = \mathbb{1}$ and $\sum_{l>M_{r-1}} \hat{P}_{r-1,l} = P_{r-1,M_{r-1}+1}$.

Writing out the recursion (using that $\|P_{r,k-1}|\psi_i\rangle\| \leq \|P_{r,k-1}|\psi_{i+1}\rangle\|$ as seen in Equation 3.2) results in

$$\begin{aligned}
&\sum_{i=0}^{q-1} \sqrt{\frac{M_{r-1}}{N}\|P_{r,k-1}|\psi_i\rangle\|^2 + \|P_{r-1,M_{r-1}+1}|\psi_i\rangle\|^2} \\
&\leq \sum_{i=0}^{q-1} \left(\sqrt{\frac{M_{r-1}}{N}}\|P_{r,k-1}|\psi_i\rangle\| + \|P_{r-1,M_{r-1}+1}|\psi_q\rangle\| \right) \\
&= \sum_{i_1=0}^{q-1} \left(\sum_{i_2=0}^{i_1} \left(\frac{M_{r-1}}{N}\|P_{r,k-2}|\psi_{i_2}\rangle\| + \sqrt{\frac{M_{r-1}}{N}}\|P_{r-1,M_{r-1}+1}|\psi_q\rangle\| \right) + \|P_{r-1,M_{r-1}+1}|\psi_q\rangle\| \right) \\
&= \sum_{0\leq i_k<\cdots<i_2<i_1\leq q-1} \left(\sqrt{\frac{M_{r-1}}{N}}\right)^k + \sum_{j=1}^{k} \sum_{0\leq i_j<\cdots<i_2<i_1\leq q-1} \left(\sqrt{\frac{M_{r-1}}{N}}\right)^{j-1} \|P_{r-1,M_{r-1}+1}|\psi_q\rangle\|.
\end{aligned} \tag{4.11}$$

Just as in the non-iterative case we bound $\sum_{0\leq i_k<\cdots<i_2<i_1\leq q-1} \leq \frac{q^k}{k!} \leq \left(\frac{eq}{k}\right)^k$ to obtain

$$\begin{aligned}
&\sum_{0\leq i_k<\cdots<i_2<i_1\leq q-1} \left(\sqrt{\frac{M_{r-1}}{N}}\right)^k + \sum_{j=1}^{k} \sum_{0\leq i_j<\cdots<i_2<i_1\leq q-1} \left(\sqrt{\frac{M_{r-1}}{N}}\right)^{j-1} \|P_{r-1,M_{r-1}+1}|\psi_q\rangle\| \\
&\leq \left(\frac{eq}{k}\right)^k \left(\sqrt{\frac{M_{r-1}}{N}}\right)^k + \sum_{j=1}^{k} \frac{q^j}{j!} \left(\sqrt{\frac{M_{r-1}}{N}}\right)^{j-1} \|P_{r-1,M_{r-1}+1}|\psi_q\rangle\| \\
&= \left(\frac{eq\sqrt{M_{r-1}}}{k\sqrt{N}}\right)^k + \sum_{j=1}^{k} \frac{q}{j} \frac{\left(q\sqrt{\frac{M_{r-1}}{N}}\right)^{j-1}}{(j-1)!} \|P_{r-1,M_{r-1}+1}|\psi_q\rangle\| \\
&\leq \left(\frac{eq\sqrt{M_{r-1}}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{M_{r-1}}{N}}} q\|P_{r-1,M_{r-1}+1}|\psi_q\rangle\|,
\end{aligned} \tag{4.12}$$

where in the final inequality the Taylor series of $e^{q\sqrt{\frac{M_{r-1}}{N}}}$ is used. □

4.1.1. Proof of Theorem 4.1.1

With the help of these Lemma's we can prove Theorem 4.1.1. The connection between Lemma 4.1.3 and Theorem 4.1.1 is to find the right M_{r-1} that minimizes Equation 4.8 to get the tightest upper bound on the quantum query solvability. To do this however, we need to know $\|P_{r-1,k}|\psi_i\rangle\|$, which creates a recursion that ends at $P_{2,k}$ and M_1. When we have found these values, we can inductively find $P_{r,k}$ and M_{r-1} for all other $r > 2$.

For clarity we will not only prove the base case $r = 2$, but also derive the solution for $r = 3$ to get an better understanding of the inductive process.

- Our choice for M_1 is clear, as we try to minimize the following function of M_1.

$$\begin{aligned}\|P_{2,k}|\psi_q\rangle\| &\leq \left(\frac{eq\sqrt{M_1}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{M_1}{N}}} q\|P_{1,M_1+1}|\psi_q\rangle\| \\ &\leq \left(\frac{eq\sqrt{M_1}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{M_1}{N}}} q\mathbb{1}_{[M_1+1\leq q]}, \end{aligned} \tag{4.13}$$

  where we used that there are at most q tuples in the database after q queries and thus $\|P_{1,M_1+1}|\psi_q\rangle\| = 0$ if $M_1 + 1 > q$. Whenever $M_1 < q$ it holds that $\mathbb{1}_{[M_1+1\leq q]} = 1$, but both $\left(\frac{eq\sqrt{M_1}}{k\sqrt{N}}\right)^k$ and $e^{q\sqrt{\frac{M_1}{N}}}$ are increasing in M_1. So we minimize our bound on $\|P_{2,k}|\psi_q\rangle\|$ by setting $M_1 = q$, in which case $\|P_{2,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{q}}{k\sqrt{N}}\right)^k = \left(\frac{eq^{3/2}}{k\sqrt{N}}\right)^k$.

- For $r = 3$, we have

$$\|P_{3,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{M_2}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{M_2}{N}}} q\left(\frac{eq^{3/2}}{M_2\sqrt{N}}\right)^{M_2}, \tag{4.14}$$

  where we substituted the bound we just computed for $\|P_{2,k}|\psi_q\rangle\|$ into Equation 4.12. Since we want to minimize this expression for M_2, we need $\frac{eq^{3/2}}{\sqrt{N}} < M_2$, as otherwise $\left(\frac{eq^{3/2}}{M_2\sqrt{N}}\right)^{M_2}$ will be at least 1. As a result we choose $M_2 = \frac{2eq^{3/2}}{\sqrt{N}}$. Even though this is not the actual analytical optimum for M_2, this approximation will simplify the analysis. This simplification does not impact the resulting query complexity, as we will show in a moment.

  Let us consider the case that $q > \sqrt{kN}$. To make sure that in this case it still holds that $\frac{eq\sqrt{M_2}}{k\sqrt{N}} \leq 1$ such that our bound on $\|P_{3,k}|\psi_q\rangle\|$ does not simply result in 1, we find that $M_2 < \frac{k}{e^2} < k$. Combining this with our previously found fact that $\frac{eq^{3/2}}{\sqrt{N}} < M_2$ this results in $k > \frac{\sqrt{2}e^{3/2}q^{7/4}}{N^{3/4}} > k^{7/8}N^{1/8}$ and thus $k > N$. Since this would imply that $q > N$ we instead conclude that in the case of $q > \sqrt{kN}$ our bound on $\|P_{3,k}|\psi_q\rangle\|$ becomes trivial, i.e. 1.

  In the case that $q \leq \sqrt{kN}$ this implies that $q\sqrt{\frac{M_2}{N}} \leq 2ek^{7/8}N^{1/8}$ and so we can bound

$$\begin{aligned} e^{q\sqrt{\frac{M_2}{N}}} q\left(\frac{eq^{3/2}}{M_2\sqrt{N}}\right)^{M_2} &\leq e^{2ek^{7/8}N^{1/8}} q\left(\frac{1}{2}\right)^{M_2} \\ &\leq e^{2ek^{7/8}N^{1/8}} N\left(\frac{1}{2}\right)^{M_2} \\ &\leq \left(\frac{1}{2}\right)^{-8k^{7/8}N^{1/8}} N\left(\frac{1}{2}\right)^{M_2}. \end{aligned} \tag{4.15}$$

To further simplify this, we express N as a power of 2. As r is just a constant, we can state $N = \mathcal{O}\left(2^{N^{1/2^r}}\right)$. Since $\left(\frac{1}{2}\right)^{M_2}$ is decreasing in M_2, we need to find a lower bound on M_2 to upper bound $\left(\frac{1}{2}\right)^{M_2}$. We force this by updating our choice of M_2 to $M_2 = \max\left\{\frac{2eq^{3/2}}{\sqrt{N}}, 10k^{7/8}N^{1/8}\right\}$, in which case it holds that

$$\begin{aligned} e^{q\sqrt{\frac{M_2}{N}}} q \left(\frac{eq^{3/2}}{M_2\sqrt{N}}\right)^{M_2} &\leq \left(\frac{1}{2}\right)^{-8k^{7/8}N^{1/8}} N \left(\frac{1}{2}\right)^{M_2} \\ &\leq \left(\frac{1}{2}\right)^{-8k^{7/8}N^{1/8}} \mathcal{O}\left(2^{N^{1/8}}\right) \left(\frac{1}{2}\right)^{10k^{7/8}N^{1/8}} \\ &\leq \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right). \end{aligned} \tag{4.16}$$

Since we changed M_2, it no longer holds that $M_2 = \frac{2eq^{3/2}}{\sqrt{N}}$, but instead it we bound it by

$$\sqrt{M_2} = \sqrt{\max\left\{\frac{2eq^{3/2}}{\sqrt{N}}, 10k^{7/8}N^{1/8}\right\}} \leq \sqrt{\frac{2eq^{3/2}}{\sqrt{N}}} + \sqrt{10k^{7/8}N^{1/8}}. \tag{4.17}$$

By noting that $\frac{2eq^{3/2}}{\sqrt{N}} < 10k^{7/8}N^{1/8}$ whenever $q < \frac{10^{2/3}k^{7/12}N^{5/12}}{(2e)^{2/3}}$ the final bound becomes

$$\begin{aligned} \|P_{3,k}|\psi_q\rangle\| &\leq \left(\frac{eq\sqrt{\frac{2eq^{3/2}}{\sqrt{N}}}}{k\sqrt{N}} + \frac{e\frac{10^{2/3}k^{7/12}N^{5/12}}{(2e)^{2/3}}\sqrt{10k^{7/8}N^{1/8}}}{k\sqrt{N}}\right)^k + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right) \\ &\leq \left(\frac{\sqrt{2}e^{3/2}q^{7/4}}{kN^{3/4}} + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/48}\right)\right)^k + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right). \end{aligned} \tag{4.18}$$

Even though we severely reduce the tightness of our upper bound on the quantum query solvability in the case that $q < \frac{10^{2/3}k^{7/12}N^{5/12}}{(2e)^{2/3}}$ by adding this $\mathcal{O}\left(\left(\frac{k}{N}\right)^{1/48}\right)$ term, we note that this term is independent of q and as a result will not affect the resulting quantum query complexity.

For completeness, we do need to show that for the new value of M_2 it still holds that $q\sqrt{\frac{M_2}{N}} \leq 2ek^{7/8}N^{1/8}$:

$$\begin{aligned} q\sqrt{\frac{M_2}{N}} &\leq \frac{10^{7/6}k^{49/48}}{(2e)^{2/3}N^{1/48}} \\ &\leq \frac{10^{7/6}k^{7/8}N^{1/8}}{(2e)^{2/3}} \\ &\leq 2ek^{7/8}N^{1/8}, \end{aligned} \tag{4.19}$$

where in the second inequality we used that $k \leq N$. We can assume this inequality, as otherwise $\left(\frac{k}{N}\right)^{1/48}$ will be larger than 1.

- The procedure for $r = 4$ can be found in Appendix D if there is need for an extra example.

Now assume that the induction hypothesis holds, then by Lemma 4.1.3 it holds that

$$\|P_{r+1,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{M_r}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{M_r}{N}}} q \left(\left(\frac{\gamma(r) q^{(2^r-1)/2^{r-1}}}{M_r N^{(2^{r-1}-1)/2^{r-1}}} + \mathcal{O}\left(\left(\frac{M_r}{N}\right)^{1/(2^{r-1}-1)2^{r+1}}\right)\right)^{M_r} + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r} N^{1/2^r}}\right)\right). \tag{4.20}$$

Once again we need $\gamma(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}} < M_r < \frac{k}{(qe)^2}$, which in the case of $q > \sqrt{kN}$ results in $k > k^{(2^r-1)/2^r} N^{1/2^{r-1}}$, which implies $k > N$. As this results in $q > N$ with arrive at a contradiction and conclude that in this case our bound is trivial, i.e. 1.

We choose $M_r = 2\gamma(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}}$, which in the case that $q \leq \sqrt{kN}$ yields

$$\begin{aligned}
&e^{q\sqrt{\frac{M_r}{N}}} q \left(\left(\frac{\gamma(r) q^{(2^r-1)/2^{r-1}}}{M_r N^{(2^{r-1}-1)/2^{r-1}}} + \mathcal{O}\left(\left(\frac{M_r}{N}\right)^{1/(2^{r-1}-1)2^{r+1}}\right)\right)^{M_r} + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r} N^{1/2^r}}\right)\right) \\
&\leq e^{2ek^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} N \left(\left(\frac{1}{2} + \mathcal{O}\left(\left(\frac{M_r}{N}\right)^{1/(2^{r-1}-1)2^{r+1}}\right)\right)^{M_r} \right. \\
&\quad \left. + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r} N^{1/2^r}}\right)\right) \\
&\leq \left(\frac{1}{2}\right)^{-8k^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} \mathcal{O}\left(2^{N^{1/2^{r+1}}}\right) \left(\mathcal{O}\left(\frac{1}{2}\right)^{M_r} + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r} N^{1/2^r}}\right)\right),
\end{aligned} \tag{4.21}$$

where in the last inequality we used that $M_r \leq N$. By updating

$M_r = \max\left\{2\gamma(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}}, 10k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}\right\}$, we find

$$\begin{aligned}
&\left(\frac{1}{2}\right)^{-8k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}} \mathcal{O}\left(2^{N^{1/2^{r+1}}}\right)\left(\mathcal{O}\left(\frac{1}{2}\right)^{M_r} + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r}N^{1/2^r}}\right)\right) \\
&\leq \left(\frac{1}{2}\right)^{-8k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}} \mathcal{O}\left(2^{N^{1/2^{r+1}}}\right)\left(\mathcal{O}\left(\frac{1}{2}\right)^{10k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}}\right. \\
&\quad \left. + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r}N^{1/2^r}}\right)\right) \\
&\leq \mathcal{O}\left(2^{-k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}}\right). \qquad (4.22)
\end{aligned}$$

Lastly $2\gamma(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}} \leq 10k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}$ implies
$q \leq \frac{10^{2^{r-1}/(2^r-1)}k^{(2(2^r-1)+1)/4(2^r-1)}N^{(2(2^r-1)-1)/4(2^r-1)}}{(2e)^{(2^r-2)/(2^r-1)}}$, which results in the final bound being

$$\begin{aligned}
\|P_{r+1,k}|\psi_q\rangle\| &\leq \left(\frac{eq\sqrt{2\gamma(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}}}}{k\sqrt{N}}\right. \\
&\quad \left. + \frac{\frac{10^{2^{r-1}/(2^r-1)}k^{(2(2^r-1)+1)/4(2^r-1)}N^{(2(2^r-1)-1)/4(2^r-1)}}{(2e)^{(2^r-2)/(2^r-1)}}\sqrt{10k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}}}{k\sqrt{N}}\right)^k \\
&\quad + \mathcal{O}\left(2^{-k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}}\right) \\
&\leq \left(\frac{\gamma(r+1)q^{(2^{r+1}-1)/2^r}}{kN^{(2^r-1)/2^r}} + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/(2^r-1)2^{r+2}}\right)\right)^k + \\
&\quad + \mathcal{O}\left(2^{-k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}}\right) \qquad (4.23)
\end{aligned}$$

and the induction is completed.

For the new value of M_r our bound on $q\sqrt{\frac{M_r}{N}}$ is still correct, since

$$\begin{aligned}
q\sqrt{\frac{M_r}{N}} &\leq \frac{10^{(2(2^r-1)+1)/2(2^r-1)}k^{((2^{r-1}-1)2^{r+1}+1)/(2^{r-1}-1)2^{r+1}}}{(2e)^{(2^r-2)/(2^r-1)}N^{1/(2^{r-1}-1)2^{r+1}}} \\
&\leq \frac{10^{(2(2^r-1)+1)/2(2^r-1)}k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}}{(2e)^{6/7}} \\
&\leq 2ek^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}. \qquad (4.24)
\end{aligned}$$

By applying Lemma 3.1.3 to the above theorem we derive a bound on the quantum query solvability of the Collision problem.

Corollary 4.1.4. *Any quantum algorithm A making q queries to a random oracle $F : \{0,1\}^m \to \{0,1\}^n$ finds k distinct r-collisions, for $k \leq \sqrt{\frac{N}{r}}$ and constant $r \geq 2$, with probability at most*

$$\left(\left(\mathcal{O}\left(\frac{q^{(2^r-1)/2^{r-1}}}{kN^{(2^{r-1}-1)/2^{r-1}}}\right)+\mathcal{O}\left(\left(\frac{k}{N}\right)^{1/(2^{r-2}-1)2^r}\right)\right)^k + \mathcal{O}\left(2^{-k^{(2^{r-1}-1)/2^{r-1}}N^{1/2^{r-1}}}\right)+k\sqrt{\frac{r}{N}}\right)^2. \tag{4.25}$$

As a result, any quantum algorithm A needs to make $\Omega\left(k^{2^{r-1}/(2^r-1)}N^{(2^{r-1}-1)/(2^r-1)}\right)$ queries to find k distinct r-collisions with constant probability.

For $k = 1$ matching upper bounds on the query complexity are given by [Hos+18; LZ18], showing the tightness of the bound.

4.2. General iterative relations

Now that we have seen the technique for $\mathsf{R}^r_{\mathsf{Col}}$, we can generalize it to any R^r in general. Without loss of generality we can assume R^r to be maximally reduced. Whereas in the case of $\mathsf{R}^r_{\mathsf{Col}}$ we found that $|G_{\mathsf{R}^r_{\mathsf{Col}},1,l}| = l$, in the general case we just keep it uncomputed as $|G_{\mathsf{R}^r,1,l}|$. The general case of Lemma 4.1.3 then states

Theorem 4.2.1. *Let $P_{r,k}$ be the projector which projects onto the database containing at least k distinct r-tuples satisfying the maximally reduced relation R^r. Then for any $M_{r-1} \leq N$, $k \geq 1$ and constant $r \geq r_0$ it holds that*

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{|G_{\mathsf{R}^r,1,M_{r-1}}|}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{|G_{\mathsf{R}^r,1,M_{r-1}}|}{N}}}q\|P_{r-1,M_{r-1}+1}|\psi_q\rangle\|, \tag{4.26}$$

where $r_0 = \min\{r : \mathsf{R}^r$ is not the trivial relation$\}$.

Proof. Just like l, $|G_{\mathsf{R}^r,1,l}|$ takes on values in $\{0,...,l\}$ and as a result the same trick of introducing M_{r-1} as in Lemma 4.1.3 can be applied to bound

$$\begin{aligned}
\|P_{r,k}|\psi'_{i,\bar{k}_r,(k-1)_r}\rangle\| &\leq \sum_{i=0}^{q-1}\sqrt{\sum_{l=0}^{N}\frac{|G_{\mathsf{R}^r,1,l}|}{N}\|\hat{P}_{r-1,l}P_{r,k-1}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|^2} \\
&= \sum_{i=0}^{q-1}\left(\sum_{l=0}^{M_{r-1}}\frac{|G_{\mathsf{R}^r,1,l}|}{N}\|\hat{P}_{r-1,l}P_{r,k-1}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|^2\right. \\
&\quad \left.+\left(\sum_{l=M_{r-1}+1}^{N}\frac{|G_{\mathsf{R}^r,1,l}|}{N}\|\hat{P}_{r-1,l}P_{r,k-1}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|^2\right)^{1/2}\right) \\
&\leq \sum_{i=0}^{q-1}\sqrt{\frac{|G_{\mathsf{R}^r,1,M_{r-1}}|}{N}\|P_{r,k-1}|\psi_i\rangle\|^2+\|P_{r-1,M_{r-1}+1}|\psi_i\rangle\|^2},
\end{aligned} \tag{4.27}$$

where in the final inequality we used that $|G_{\mathsf{R}^r,1,l}|$ is increasing in l. By the same calculation as in the proof of Lemma 4.1.3 it holds that

$$\sum_{i=0}^{q-1} \sqrt{\frac{|G_{\mathsf{R}^r,1,M_{r-1}}|}{N}\|P_{r,k-1}|\psi_i\rangle\|^2 + \|P_{r-1,M_{r-1}+1}|\psi_i\rangle\|^2} \\ \leq \left(\frac{eq\sqrt{|G_{\mathsf{R}^r,1,M_{r-1}}|}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{|G_{\mathsf{R}^r,1,M_{r-1}}|}{N}}} q\|P_{r-1,M_{r-1}+1}|\psi_q\rangle\|. \tag{4.28}$$

□

One interesting observation is that Theorem 3.2.3 actually follows from the above theorem. Since for non-iterative relations $|G_{\mathsf{R}^r,1,M_{r-1}}| = |G_{\mathsf{R}^r,1}|$ is independent of M_{r-1}, we can just choose $M_{r-1} = M$ such that $\|P_{r-1,M_{r-1}+1}|\psi_q\rangle\| = 0$. This then precisely results in

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{|G_{\mathsf{R}^r,1}|}}{k\sqrt{N}}\right)^k. \tag{4.29}$$

4.3. Collision problem for parallel queries

Unlike in the non-iterative case, we have not succeeded in formalising a bound on $\|P_{r,k}|\psi_q\rangle\|$ in the event of p-parallel queries for any R^r. We will however show the technique for the case of $\mathsf{R}^r_{\mathsf{Col}}$ and provide a few applications of the technique to other relations. Recall Equation 3.19 which also holds for iterative relations:

$$\|P_{r,k}(\mathsf{CFO}|\psi_{i,\overline{k}_r}\rangle)\|\|(\mathbb{I} - P_{r,k})|\psi_i\rangle\| \leq \sum_{j=1}^{\min\{p,k\}} \|P_{r,k}\mathsf{CFO}|\psi_{i,\overline{k}_r,(k-j)_r}\rangle\|\|P_{r,k-j}(\mathbb{I} - P_{r,k})|\psi_i\rangle\|. \tag{4.30}$$

To make the analysis clearer, we bound $\min\{p,r\} \leq r$ and $\min\{p,k\} \leq k$ right away, reducing the tightness of our bound in the cases that $p \leq r$ and $p \leq k$.

Theorem 4.3.1. *Let $P_{r,k}$ be the projector which projects onto the database containing at least k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Col}}$, as defined in Definition 3.1.1. Also define $\gamma'(r) = 2^{(2^{r-2}-1)/2^{r-2}} e^{3(2^{r-1}-1)/2^{r-1}} r!$. Then for any $M_{r-1} \leq N$, $k \leq N$, constant $r \geq 2$ and parallel query size p it holds that*

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{\gamma'(r)q^{(2^r-1)/2^{r-1}}p}{kN^{(2^{r-1}-1)/2^{r-1}}} + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/(2^{r-1}-1)2^{r+1}}\right)\right)^k \\ + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r}N^{1/2^r}}\right). \tag{4.31}$$

Proof. Instead of just distinguishing the different number $(r-1)$-tuples satisfying $\mathsf{R}^{r-1}_{\mathsf{Col}}$, we distinguish these numbers for all $(r-t)$-tuples satisfying $\mathsf{R}^{r-t}_{\mathsf{Col}}$, resulting in the parallel version of Equation 4.5.

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r}\rangle\|^2 = \sum_{l_0,\dots,l_{r-1}=0}^{N} \|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r,\hat{l}_0,\dots,\hat{l}_{r-1}}\rangle\|^2 \|\hat{P}_{0,l_0}\cdots\hat{P}_{r-1,l_{r-1}}|\psi_{i,\overline{k}_r,(k-j)_r}\rangle\|^2, \tag{4.32}$$

where we have extended the abbreviation $|\psi_{i,\overline{k}_r,(k-j)_r\hat{l}_{r-t}}\rangle = \frac{\hat{P}_{r-t,l_{r-t}}|\psi_{i,\overline{k}_r,(k-j)_r}\rangle}{\|\hat{P}_{r-t,l_{r-t}}|\psi_{i,\overline{k}_r,(k-j)_r}\rangle\|}$. Analogous to the proofs of Lemma 3.4.1 and Lemma 4.1.2 we bound

$$\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r,\hat{l}_0,\dots,\hat{l}_{r-1}}\rangle\| \leq \sum_{t_1,\dots,t_j=1}^{r} \left(\sqrt{\frac{l_{r-t_1}(ep)^{t_1}}{(t_1N)^{t_1}}}\cdots\sqrt{\frac{l_{r-t_j}(ep)^{t_j}}{(t_jN)^{t_j}}}\right). \tag{4.33}$$

Combining these findings with Equation 3.19 yields

$$\begin{aligned}\|P_{r,k}|\psi_q\rangle\| \leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{t_1,\dots,t_j=1}^{r}\Bigg(\sum_{l_1,\dots,l_{r-1}=0}^{N}\frac{l_{r-t_1}(ep)^{t_1}}{(t_1N)^{t_1}}\cdots\frac{l_{r-t_j}(ep)^{t_j}}{(t_jN)^{t_j}}\\ \cdot\|\hat{P}_{1,l_1}\cdots\hat{P}_{r-1,l_{r-1}}P_{r,k-j}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|^2\Bigg)^{1/2}.\end{aligned} \tag{4.34}$$

Here the sum over l_0 has been omitted, since the number of distinct 0-tuples in the database satisfying any R^r is always maximal, and thus $l_0 = |G_{\mathsf{R}^r_{\mathsf{Col}},r-1,l_0}| = N$.

We change the perspective of how we decide to view the sum over $t_1,\dots,t_j$. Instead of letting $t_1,\dots,t_j$ sum over values in $[r]$, we count for each value in $[r]$ how many t_j's take on this value. This does require an extra binomial coefficient to correct for the permutations.

As before we first present examples for the cases of $r = 2, 3$.

$$\begin{aligned}\|P_{2,k}|\psi_q\rangle\| &\leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{t_1,\dots,t_j=1}^{2}\sqrt{\sum_{l_1=0}^{N}\frac{l_{r-t_1}(ep)^{t_1}}{(t_1N)^{t_1}}\cdots\frac{l_{r-t_j}(ep)^{t_j}}{(t_jN)^{t_j}}\|\hat{P}_{1,l_1}P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2}\\ &= \sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{\#(t=1)=0}^{j}\binom{j}{\#(t=1)}\Bigg(\sum_{l_1=0}^{N}\left(\frac{l_1ep}{N}\right)^{\#(t=1)}\left(\frac{N(ep)^2}{(2N)^2}\right)^{j-\#(t=1)}\\ &\quad\cdot\|\hat{P}_{1,l_1}P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2\Bigg)^{1/2},\end{aligned} \tag{4.35}$$

where we have used that $l_0 = N$. Just as in the sequential case we now introduce M_1, resulting in

$$\sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{\#(t=1)=0}^{j}\binom{j}{\#(t=1)}\left(\sum_{l_1=0}^{N}\left(\frac{l_1 ep}{N}\right)^{\#(t=1)}\left(\frac{N(ep)^2}{(2N)^2}\right)^{j-\#(t=1)}\right.$$
$$\left.\cdot\,\|\hat{P}_{1,l_1}P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2\right)^{1/2}$$
$$=\sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{\#(t=1)=0}^{j}\binom{j}{\#(t=1)}\left(\sum_{l_1=0}^{M_1}\left(\frac{l_1 ep}{N}\right)^{\#(t=1)}\left(\frac{N(ep)^2}{(2N)^2}\right)^{j-\#(t=1)}\right.$$
$$\cdot\,\|\hat{P}_{1,l_1}P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2+\sum_{l_1=M_1+1}^{N}\left(\frac{l_1 ep}{N}\right)^{\#(t=1)}\left(\frac{N(ep)^2}{(2N)^2}\right)^{j-\#(t=1)}$$
$$\left.\cdot\,\|\hat{P}_{1,l_1}P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2\right)^{1/2}. \tag{4.36}$$

We make the following important observation. The term

$$\sum_{\#(t=1)=0}^{j}\binom{j}{\#(t=1)}\sum_{l_1=M_1+1}^{N}\left(\frac{l_1 ep}{N}\right)^{\#(t=1)}\left(\frac{N(ep)^2}{(2N)^2}\right)^{j-\#(t=1)}\|\hat{P}_{1,l_1}P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2 \tag{4.37}$$

serves as our upper bound for the expression (see Equation 4.32)

$$\sum_{l_1=M_1+1,l_2,\ldots,l_{r-1}=0}^{N}\|P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r,\hat{l}_1,\ldots,\hat{l}_{r-1}}\rangle\|^2\|\hat{P}_{1,l_1}\cdots\hat{P}_{r-1,l_{r-1}}|\psi_{i,\overline{k}_r,(k-j)_r}\rangle\|^2$$
$$=\|P_{1,M_1+1}P_{r,k}|\psi'_{i,\overline{k}_r,(k-j)_r}\rangle\|^2$$
$$\leq\|P_{1,M_1+1}\|. \tag{4.38}$$

By instead using this tighter bound, we arrive at

$$\begin{aligned}\|P_{2,k}|\psi_q\rangle\| &\leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\left(\sum_{\#(t=1)=0}^{j}\binom{j}{\#(t=1)}\left(\sum_{l_1=0}^{M_1}\left(\frac{l_1 ep}{N}\right)^{\#(t=1)}\right.\right.\\ &\quad\left.\left.\cdot\left(\frac{N(ep)^2}{(2N)^2}\right)^{j-\#(t=1)}\|\hat{P}_{1,l_1}P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2\right)^{1/2}+\|P_{1,M_1+1}|\psi_q\rangle\|\right)\\ &\leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\left(\sum_{\#(t=1)=0}^{j}\binom{j}{\#(t=1)}\left(\left(\frac{M_1 ep}{N}\right)^{\#(t=1)}\right.\right.\\ &\quad\left.\left.\cdot\left(\frac{N(ep)^2}{(2N)^2}\right)^{j-\#(t=1)}\|P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2\right)^{1/2}+\|P_{1,M_1+1}|\psi_q\rangle\|\right).\end{aligned} \tag{4.39}$$

Since $\|P_{1,M_1+1}|\psi_q\rangle\| \leq \mathbb{1}_{[M_1+1\leq qp]}$, we set $M_1 = qp$ to minimize the above expression. Since this results in $\frac{M_1 ep}{N} > \frac{N(ep)^2}{(2N)^2}$ and $\|P_{1,M_1+1}|\psi_q\rangle\| = 0$ we can bound

$$\begin{aligned}\|P_{2,k}|\psi_q\rangle\| &\leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\left(\sum_{\#(t=1)=0}^{j}\binom{j}{\#(t=1)}\left(\left(\frac{M_1 ep}{N}\right)^{\#(t=1)}\right.\right.\\ &\quad\left.\left.\cdot\left(\frac{N(ep)^2}{(2N)^2}\right)^{j-\#(t=1)}\|P_{2,k-j}(\mathbb{I}-P_{2,k})|\psi_i\rangle\|^2\right)^{1/2}+\|P_{1,M_1+1}|\psi_q\rangle\|\right)\\ &= \sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{\#(t=1)=0}^{j}\binom{j}{\#(t=1)}\sqrt{\left(\frac{eqp^2}{N}\right)^{j}}\|P_{2,k-j}|\psi_i\rangle\|\\ &= \sum_{i=0}^{q-1}\sum_{j=1}^{k}2^j\sqrt{\left(\frac{eqp^2}{N}\right)^{j}}\|P_{2,k-j}|\psi_i\rangle\|\\ &\leq \left(\frac{2eq\sqrt{eqp^2}}{k\sqrt{N}}\right)^{k} = \left(\frac{2e^{3/2}q^{3/2}p}{k\sqrt{N}}\right)^{k},\end{aligned} \tag{4.40}$$

where for the final inequality we have used Lemma 3.4.2.

The $r = 2$ case was relatively simple, as $\|P_{1,M_1+1}|\psi_q\rangle\|$ results in 0 for our choice of M_1.

That is why we also show the $r = 3$ case.

$$\begin{aligned}\|P_{3,k}|\psi_q\rangle\| \leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{t_1,\ldots,t_j=1}^{3}\Bigg(\sum_{l_1,l_2=0}^{N}\frac{l_{r-t_1}(ep)^{t_1}}{(t_1N)^{t_1}}\cdots\frac{l_{r-t_j}(ep)^{t_j}}{(t_jN)^{t_j}} \\ \cdot\|\hat{P}_{1,l_1}\hat{P}_{2,l_2}P_{3,k-j}(\mathbb{I}-P_{3,k})|\psi_i\rangle\|^2\Bigg)^{1/2} \\ \leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{\#(t=1)=0}^{j}\sum_{\#(t=2)=0}^{j-\#(t=1)}\binom{j}{\#(t=1)}\binom{j-\#(t=1)}{\#(t=2)} \\ \cdot\Bigg(\sum_{l_1,l_2=0}^{N}\left(\frac{l_2ep}{N}\right)^{\#(t=1)}\left(\frac{l_1(ep)^2}{(2N)^2}\right)^{\#(t=2)}\left(\frac{N(ep)^3}{(3N)^3}\right)^{j-\#(t=1)-\#(t=2)} \\ \cdot\|\hat{P}_{1,l_1}\hat{P}_{2,l_2}P_{3,k-j}(\mathbb{I}-P_{3,k})|\psi_i\rangle\|^2\Bigg)^{1/2}. \end{aligned} \tag{4.41}$$

By applying the same trick as in Equation 4.38 after introducing M_1, M_2 we find

$$\begin{aligned}\|P_{3,k}|\psi_q\rangle\| \leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{\#(t=1)=0}^{j}\sum_{\#(t=2)=0}^{j-\#(t=1)}\binom{j}{\#(t=1)}\binom{j-\#(t=1)}{\#(t=2)} \\ \cdot\Bigg(\sum_{l_1,l_2=0}^{N}\left(\frac{l_2ep}{N}\right)^{\#(t=1)}\left(\frac{l_1(ep)^2}{(2N)^2}\right)^{\#(t=2)}\left(\frac{N(ep)^3}{(3N)^3}\right)^{j-\#(t=1)-\#(t=2)} \\ \cdot\|\hat{P}_{1,l_1}\hat{P}_{2,l_2}P_{3,k-j}(\mathbb{I}-P_{3,k})|\psi_i\rangle\|^2\Bigg)^{1/2} \\ \leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\Bigg(\sum_{\#(t=1)=0}^{j}\sum_{\#(t=2)=0}^{j-\#(t=1)}\binom{j}{\#(t=1)}\binom{j-\#(t=1)}{\#(t=2)} \\ \cdot\Bigg(\left(\frac{M_2ep}{N}\right)^{\#(t=1)}\left(\frac{M_1(ep)^2}{(2N)^2}\right)^{\#(t=2)}\left(\frac{N(ep)^3}{(3N)^3}\right)^{j-\#(t=1)-\#(t=2)} \\ \cdot\|P_{3,k-j}(\mathbb{I}-P_{3,k})|\psi_i\rangle\|^2\Bigg)^{1/2} + \|P_{2,M_2+1}|\psi_i\rangle\| + \|P_{1,M_1+1}|\psi_i\rangle\|\Bigg). \end{aligned} \tag{4.42}$$

We again have to set $M_1 = qp$. Like in the sequential case, instead of finding the optimal value for M_2, we settle for M_2 such that $\|P_{2,M_2+1}|\psi_i\rangle\| \leq \left(\frac{1}{2}\right)^{M_2}$. From our bound on $\|P_{2,k}|\psi_q\rangle\|$ we derive that this is satisfied whenever $M_2 = \frac{4e^{3/2}q^{3/2}p}{\sqrt{N}}$. These values of M_1, M_2 result in

$\frac{M_2 ep}{N} > \frac{M_1 (ep)^2}{(2N)^2} > \frac{N(ep)^3}{(3N)^3}$ and as a consequence we can bound

$$\begin{aligned}
\|P_{3,k}|\psi_q\rangle\| \leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} \Bigg(& \sum_{\#(t=1)=0}^{j} \sum_{\#(t=2)=0}^{j-\#(t=1)} \binom{j}{\#(t=1)} \binom{j-\#(t=1)}{\#(t=2)} \\
& \cdot \Bigg(\left(\frac{M_2 ep}{N}\right)^{\#(t=1)} \left(\frac{M_1 (ep)^2}{(2N)^2}\right)^{\#(t=2)} \left(\frac{N(ep)^3}{(3N)^3}\right)^{j-\#(t=1)-\#(t=2)} \\
& \cdot \|P_{3,k-j}(\mathbb{I} - P_{3,k})|\psi_i\rangle\|^2 \Bigg)^{1/2} + \|P_{2,M_2+1}|\psi_i\rangle\| + \|P_{1,M_1+1}|\psi_i\rangle\| \Bigg) \\
\leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} \Bigg(& \sum_{\#(t=1)=0}^{j} \sum_{\#(t=2)=0}^{j-\#(t=1)} \binom{j}{\#(t=1)} \binom{j-\#(t=1)}{\#(t=2)} \\
& \cdot \sqrt{\left(\frac{M_2 ep}{N}\right)^{j}} \|P_{3,k-j}|\psi_i\rangle\| + \|P_{2,M_2+1}|\psi_i\rangle\| \Bigg) \\
= \sum_{i=0}^{q-1} \sum_{j=1}^{k} & \left(3^j \sqrt{\left(\frac{M_2 ep}{N}\right)^{j}} \|P_{3,k-j}|\psi_i\rangle\| + \|P_{2,M_2+1}|\psi_i\rangle\| \right) \\
\leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} & \left(3^j \sqrt{\left(\frac{M_2 ep}{N}\right)^{j}} \|P_{3,k-j}|\psi_i\rangle\| + \|P_{2,M_2+1}|\psi_q\rangle\| \right), \qquad (4.43)
\end{aligned}$$

where we have kept the variable M_2 instead of $\frac{4e^{3/2}q^{3/2}p}{\sqrt{N}}$ for clarity. By consulting Lemma 3.4.2 we can bound this expression even further:

$$\begin{aligned}
\|P_{3,k}|\psi_q\rangle\| &\leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} \left(3^j \sqrt{\left(\frac{M_2 ep}{N}\right)^{j}} \|P_{3,k-j}|\psi_i\rangle\| + \|P_{2,M_2+1}|\psi_q\rangle\| \right) \\
&= \frac{q^k}{k!} \left(\frac{3\sqrt{M_2 ep}}{\sqrt{N}}\right)^k + \sum_{j=1}^{k} \frac{q^j}{j!} \left(\frac{3\sqrt{M_2 ep}}{\sqrt{N}}\right)^{j-1} \|P_{2,M_2+1}|\psi_q\rangle\| \\
&= \left(\frac{3eq\sqrt{M_2 ep}}{k\sqrt{N}}\right)^k + e^{3q\sqrt{\frac{M_2 ep}{N}}} q \|P_{2,M_2+1}|\psi_q\rangle\|. \qquad (4.44)
\end{aligned}$$

Just like in the sequential case, we do a case analysis for $q > \sqrt{\frac{kN}{p}}$ and $q \leq \sqrt{\frac{kN}{p}}$. If $q > \sqrt{\frac{kN}{p}}$, then $M_2 < \frac{k}{e^3} < k$ to ensure that $\frac{3eq\sqrt{M_2 ep}}{k\sqrt{N}}$ is smaller than 1. This implies however that $k > \frac{2e^{3/2}q^{3/2}p}{\sqrt{N}} > k^{3/4}(Np)^{1/4}$, which results in $k > Np$ and thus $q > \frac{N}{p}$. Since this is impossible, as $qp \leq N$ we conclude that for $\frac{3eq\sqrt{M_2 ep}}{k\sqrt{N}}$ our bound is trivial, i.e. 1.

In the case that $q \leq \sqrt{\frac{kN}{p}}$ we can bound $e^{3q\sqrt{\frac{M_2 ep}{N}}} q \| P_{2,M_2+1} |\psi_q\rangle \|$ as

$$\begin{aligned} e^{3q\sqrt{\frac{M_2 ep}{N}}} q \left(\frac{2e^{3/2}q^{3/2}p}{M_2\sqrt{N}}\right)^{M_2} &\leq e^{6ek^{7/8}N^{1/8}} N \left(\frac{1}{2}\right)^{M_2} \\ &\leq \left(\frac{1}{2}\right)^{-24k^{7/8}N^{1/8}} \mathcal{O}\left(2^{N^{1/8}}\right) \left(\frac{1}{2}\right)^{M_2} \\ &\leq \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right), \end{aligned} \tag{4.45}$$

where in the last inequality we have updated $M_2 = \max\left\{\frac{4e^{3/2}q^{3/2}p}{\sqrt{N}}, 26k^{7/8}N^{1/8}\right\}$. Since we changed M_2, it no longer holds that $M_2 = \frac{4e^{3/2}q^{3/2}p}{\sqrt{N}}$, but instead it can be bounded as

$$\sqrt{M_2} = \sqrt{\max\left\{\frac{4e^{3/2}q^{3/2}p}{\sqrt{N}}, 26k^{7/8}N^{1/8}\right\}} \leq \sqrt{\frac{4e^{3/2}q^{3/2}p}{\sqrt{N}}} + \sqrt{26k^{7/8}N^{1/8}}. \tag{4.46}$$

Also as we're increasing M_2, it still holds that $\frac{M_2 ep}{N} > \frac{M_1 (ep)^2}{(2N)^2}$. By noting that $\frac{4e^{3/2}q^{3/2}p}{\sqrt{N}} \leq 26k^{7/8}N^{1/8}$ whenever $q \leq \frac{26^{2/3}k^{7/12}N^{5/12}}{4^{2/3}ep^{2/3}}$ the final bound becomes

$$\begin{aligned} \|P_{3,k}|\psi_q\rangle\| &\leq \left(\frac{3eq\sqrt{\frac{4e^{3/2}q^{3/2}p}{\sqrt{N}}ep}}{k\sqrt{N}} + \frac{3e\frac{26^{2/3}k^{7/12}N^{5/12}}{4^{2/3}ep^{2/3}}\sqrt{26k^{7/8}N^{1/8}ep}}{k\sqrt{N}}\right)^k + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right) \\ &\leq \left(\frac{6e^{9/4}q^{7/4}p}{kN^{3/4}} + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/48}\right)\right)^k + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right). \end{aligned} \tag{4.47}$$

Continuing this inductively one can show that

$$\begin{aligned} \|P_{r,k}|\psi_q\rangle\| \leq &\left(\frac{\gamma(r)q^{(2^r-1)/2^{r-1}}p}{kN^{(2^{r-1}-1)/2^{r-1}}} + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/(2^{r-1}-1)2^{r+1}}\right)\right)^k \\ &+ \mathcal{O}\left(2^{-k^{(2^r-1)/2^r}N^{1/2^r}}\right), \end{aligned} \tag{4.48}$$

where $\gamma'(r) = 2^{(2^{r-2}-1)/2^{r-2}} e^{3(2^{r-1}-1)/2^{r-1}} r!$. This induction step is formally shown in Appendix E. □

Corollary 4.3.2. *Any quantum algorithm A making q p-parallel queries to a random oracle $F : \{0,1\}^m \to \{0,1\}^n$ finds k distinct r-collisions, for $k \leq \sqrt{\frac{N}{r}}$ and constant $r \geq 2$, with probability at most*

$$\begin{aligned} &\left(\left(\mathcal{O}\left(\frac{q^{(2^r-1)/2^{r-1}}p}{kN^{(2^{r-1}-1)/2^{r-1}}}\right) + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/(2^{r-2}-1)2^r}\right)\right)^k \right. \\ &\left. + \mathcal{O}\left(2^{-k^{(2^{r-1}-1)/2^{r-1}}N^{1/2^{r-1}}}\right) + k\sqrt{\frac{r}{N}}\right)^2. \end{aligned} \tag{4.49}$$

As a result, any quantum algorithm A needs to make $\Omega\left((k/p)^{2^{r-1}/(2^r-1)}N^{(2^{r-1}-1)/(2^r-1)}\right)$ queries to find k distinct r-collisions with constant probability.

If we substitute $k = 1, r = 2$ our resulting query complexity is consistent with the tight result of [JMW17], where they give an query complexity bound of $\Omega\left((M/p)^{2/3}\right)$ under the assumption that $M \geq \sqrt{N}$.

When comparing the number of total queries needed for constant probabilities in our sequential result (where the total number of queries is q) to our parallel result (where the total number of queries is qp), we see that the parallel case needs a factor of $p^{(2^{r-1}-1)/(2^r-1)}$ more queries to obtain constant probability of success.

4.4. Fully parallel queries for iterative relations

Just as discussed in the previous chapter, our bounds are no longer tight when we look at the fully parallel case. To bound these cases we return to Equation 4.34:

$$\|P_{r,k}|\psi_q\rangle\| \leq \sum_{i=0}^{q-1}\sum_{j=1}^{k}\sum_{t_1,\ldots,t_j=1}^{r} \cdot\sqrt{\sum_{l_1,\ldots,l_{r-1}=0}^{N}\frac{l_{t_1}(ep)^{t_1}}{(t_1N)^{t_1}}\cdots\frac{l_{t_j}(ep)^{t_j}}{(t_jN)^{t_j}}\|\hat{P}_{1,l_{r-1}}\cdots\hat{P}_{r-1,l_1}P_{r,k-j}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|^2}. \tag{4.50}$$

In the case of $q = 1$, the only value for i will be 0. As a result $\|P_{r,k-j}|\psi_i\rangle\| = \delta_{j,k}$.
In other words

$$\|P_{r,k}|\psi_q\rangle\| \leq \sum_{t_1,\ldots,t_k=1}^{r}\sqrt{\sum_{l_1,\ldots,l_{r-1}=0}^{N}\frac{l_{t_1}(ep)^{t_1}}{(t_1N)^{t_1}}\cdots\frac{l_{t_k}(ep)^{t_k}}{(t_kN)^{t_k}}\|\hat{P}_{1,l_{r-1}}\cdots\hat{P}_{r-1,l_1}|\psi_0\rangle\|^2}. \tag{4.51}$$

As in the fully parallel case $q - 1 = 0$, so the only tuples in the database after 0 queries are those of length $0 = r - r$. Thus l_t is 0 for any $t \neq r$. We conclude

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\sqrt{\frac{N(ep)^r}{(rN)^r}}\right)^k \tag{4.52}$$

and the more general case

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\sqrt{\frac{|G_{\mathsf{R}^r,r}|(ep)^r}{(rN)^r}}\right)^k. \tag{4.53}$$

Corollary 4.4.1. *Any quantum algorithm A making a single fully p-parallel query to a random oracle $F : \{0,1\}^m \to \{0,1\}^n$ finds k distinct r-collisions, for $k \leq \sqrt{\frac{N}{r}}$ and constant $r \geq 2$, with probability at most*

$$\left(\left(\sqrt{\frac{N(ep)^r}{(rN)^r}} \right)^k + k\sqrt{\frac{r}{N}} \right)^2 . \tag{4.54}$$

As a result, any quantum algorithm A needs to make a parallel query of size at least $\Omega\left(N^{r-1/r}\right)$ to find k distinct r-collisions with constant probability.

This is equal to the classical bound [Suz+06].

5. Applications

5.1. Chain of Values problem

One of the applications of this result is to lower bound the quantum query complexity of the Chain of Values problem.

Definition 5.1.1. *An r-tuple $(x_1, y_1), ..., (x_r, y_r) \in (\{0,1\}^m \times \{0,1\}^n)^r$ satisfies the Chain of Values relation $\mathsf{R}^r_{\mathsf{CoV}}$ when:*

- $\forall i \in [r-1]$ *it holds that* $x_{i+1} = y_i$.

We see that reducing this (iterative) relation results in the trivial relation, which indeed confirms the known result that the Chain of Values problem is trivial for sequential queries. Recall from our discussion in Section 3.5 that in the fully parallel case we can no longer apply all our steps in the reduction process. As a result, $\mathsf{R}_{\mathsf{CoV}}$ is already maximally reduced in the fully parallel case after rewriting every constraint to the form $y_i = x_{i+1}$.

We are left with deriving the value $|G_{\mathsf{R}^r_{\mathsf{CoV}},r}|$. For every $(r-t)$-tuple satisfying $\mathsf{R}^{r-t}_{\mathsf{CoV}}$ there is one unique assignment of values to $w_1, ..., w_t$ such that it is completed into an r-tuple satisfying $\mathsf{R}^r_{\mathsf{CoV}}$. So we conclude that $|G_{\mathsf{R}^r_{\mathsf{CoV}},r}| = N$, resulting in the following corollary:

Corollary 5.1.2. *Any quantum algorithm A making a single fully p-parallel query to a random oracle $F : \{0,1\}^n \to \{0,1\}^n$ finds k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{CoV}}$, for $k \leq \sqrt{\frac{N}{r}}$ and constant $r \geq 2$, with probability at most*

$$\left(\left(\sqrt{\frac{N(ep)^r}{(rN)^r}} \right)^k + k\sqrt{\frac{r}{N}} \right)^2 . \tag{5.1}$$

As a result, any quantum algorithm A needs to make a parallel query of size at least $\Omega\left(N^{r-1/r}\right)$ to find k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{CoV}}$ with constant probability.

This result shows that the Chain of Values problem is still hard, even if the adversary has quantum access to the underlying function.

5.2. Multiclaw problem

Definition 5.2.1. *An r-tuple $(x_1, y_1), ..., (x_r, y_r) \in ((\bigcup_{i=1}^{r} \{0,1\}^{m_i}) \times \{0,1\}^n)^r$ satisfies the Multiclaw relation $\mathsf{R}^r_{\mathsf{Mc}}$ when:*

- $\forall i, j \in [r]$ *it holds that* $x_i \neq x_j \iff i \neq j$.
- $\forall i \in [r]$ *it holds that* $x_i \in \{0,1\}^{m_i}$.
- $\exists y \in \{0,1\}^n$ *such that* $\forall i \in [r]$ *we have* $y_i = y$.

Even though Multiclaw problem is more complex than the Collision problem, by running our reduction process we see that both relations reduce to the same relation (even in the fully parallel case), namely

- $\exists y \in \{0,1\}^n$ such that $\forall i \in [r]$ we have $y_i = y$.

Corollary 5.2.2. *Any quantum algorithm A making q p-parallel queries to random oracles $F_i : \{0,1\}^{m_i} \to \{0,1\}^n$ for $i \in [r]$ finds k distinct r-claws, for $k \leq \sqrt{\frac{N}{r}}$ and constant $r \geq 2$, with probability at most*

$$\left(\left(\mathcal{O}\left(\frac{q^{(2^r-1)/2^{r-1}}p}{kN^{(2^{r-1}-1)/2^{r-1}}}\right) + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/(2^{r-2}-1)2^r}\right)\right)^k + \mathcal{O}\left(2^{-k^{(2^{r-1}-1)/2^{r-1}}N^{1/2^{r-1}}}\right) + k\sqrt{\frac{r}{N}}\right)^2 \tag{5.2}$$

As a result, any quantum algorithm A needs to make $\Omega\left((k/p)^{2^{r-1}/(2^r-1)}N^{(2^{r-1}-1)/(2^r-1)}\right)$ queries to find k distinct r-claws with constant probability.

If we substitute $k = p = 1$ our resulting query complexity is matched by the upper bound of [Hos+18] and thus tight.

Corollary 5.2.3. *Any quantum algorithm A making a single fully p-parallel query to random oracles $H_i : \{0,1\}^{m_i} \to \{0,1\}^n$ for $i \in [r]$ finds k distinct r-claws, for $k \leq \sqrt{\frac{N}{r}}$ and constant $r \geq 2$, with probability at most*

$$\left(\left(\sqrt{\frac{N(ep)^r}{(rN)^r}}\right)^k + k\sqrt{\frac{r}{N}}\right)^2 \tag{5.3}$$

As a result, any quantum algorithm A needs to make a parallel query of size at least $\Omega\left(N^{r-1/r}\right)$ to find k distinct r-claws with constant probability.

We see from these results that in the asymptotic case finding multiclaws is just as difficult as finding collisions for any quantum adversary.

6. Conclusion

We have presented a framework which by incorporating the recently developed compressed-oracle technique and its applications [LZ18; Zha18] allows for an easier analysis of finding lower bounds on quantum query complexities. Using this method we have managed to improve upon existing quantum query lower bounds by either extending the range of parameters and/or allow parallel queries. Apart from this we have also proven the new result of hardness of the Chain of values problem in the quantum setting.

6.1. Future work

One can only say that a quantum query complexity problem is truly resolved when an algorithm has been found that matches the known lower bound. For many of our parameters and especially for the parallel case these algorithms have not yet been found. A promising candidate for an algorithm for iterative relations would be Belovs' learning-graph technique[Bel12], as this has been parallelised by [JMW17] giving optimal parallel bounds for some of our parameters. For non-iterative relations, one could research whether the algorithm by [Hos+18] allows for parallel queries and for generalisation to other problems. Since in its core this algorithm is a relative simple recursive application of Grover's algorithm [Gro96] this might be possible, albeit that this core does wear quite a complex jacket.

Perhaps a little more interesting from a theoretical point of view would be to make more use of the power of the compressed-oracle technique. As shown in [Cza+19] we do not have to restrict ourselves to uniformly random functions, but we can use the compressed oracle for any distribution. This could potentially pave the way to finding a tight quantum query complexity bound for the k-distinctness problem (or in our notation r-distinctness), which is currently one of the main open problems this field.

6.2. Summary of lower bound results

Here we summarize our lower bounds on the quantum query lower complexity for an adversary to find k distinct $r-$tuples satisfying some relation R^r with constant probability for $k \leq \sqrt{\frac{N}{r}}$, constant $r \geq 1$ and parallel query size p.

Table 6.1.: Lower bounds on the quantum query complexity for the Sum problem and the Collision problem

Relation	Sum	Collision
Parallel $q \geq kr$	$\Omega\left(k^{2/(r+1)}p^{-r/(r+1)}N^{1/(r+1)}\right)$	$\Omega\left((k/p)^{2^{r-1}/(2^r-1)}N^{(2^{r-1}-1)/(2^r-1)}\right)$
Fully parallel	$\Omega\left(N^{1/r}\right)$	$\Omega\left(N^{r-1/r}\right)$

Table 6.2.: Lower bounds on the quantum query complexity for the Chain of Values problem and the Multiclaw problem

Relation	Chain-of-Values	Multiclaw
Parallel $q \geq kr$	1	$\Omega\left((k/p)^{2^{r-1}/(2^r-1)}N^{(2^{r-1}-1)/(2^r-1)}\right)$
Fully parallel	$\Omega\left(N^{r-1/r}\right)$	$\Omega\left(N^{r-1/r}\right)$

Popular summary

Our electronics use various cryptographic systems to keep all of our data safe. Anyone who tries to get information from our data has to solve some extremely challenging problem to do so. In our internet security this problem is most often finding prime factors of some large number. In 1994 however Peter Shor discovered that this prime factor problem is not that challenging if you have access to a quantum computer. Whereas our regular computers work with bits, the tiniest possible variables which are either 0 or 1, a quantum computer works with qubits. Each such qubits can be in a superposition of 0 and 1, which means that it can be either 0, 1 or even a little of both. Using the principles of quantum mechanics, one can design algorithms that make heavy use of this property of qubits. These algorithms sometimes solve problems that we thought to be almost impossible. To make sure that all of our data stays secure even after the arrival of these quantum computers, cryptoanalysts are trying to find out which problems can be easily solved by a quantum computer and which problems remain challenging.

In this thesis we study a new technique that helps us to show if a problem is challenging to a quantum computer or not. In the problems that we sketch, an adversary has to discover some property of a function. The new technique works by creating and maintaining a quantum database that records what the adversary knows about our function. By then analyzing this database, we can derive what the adversary knows and as a result compute the hardness of our problem. To get a better understanding of this technique, we have also programmed an algorithm that simulates this quantum database on a classical computer. Since this is very difficult to do for a classical computer, we can only simulate a very small database.

Bibliography

[Amb02] Andris Ambainis. "Quantum lower bounds by quantum arguments". In: *Journal of Computer and System Sciences* 64.4 (2002), pp. 750–767.

[BC98] Vincenzo Barone and Maurizio Cossi. "Quantum calculation of molecular energies and energy gradients in solution by a conductor solvent model". In: *The Journal of Physical Chemistry A* 102.11 (1998), pp. 1995–2001.

[Bea+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. "Quantum lower bounds by polynomials". In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 778–797.

[Bel12] Aleksandrs Belovs. "Learning-graph-based quantum algorithm for k-distinctness". In: *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE. 2012, pp. 207–216.

[Ber+07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge functions". In: *ECRYPT hash workshop*. Vol. 2007. 9. Citeseer. 2007.

[Bon+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random oracles in a quantum world". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2011, pp. 41–69.

[BR93] Mihir Bellare and Phillip Rogaway. "Random oracles are practical: A paradigm for designing efficient protocols". In: *Proceedings of the 1st ACM conference on Computer and communications security*. ACM. 1993, pp. 62–73.

[BS12] Aleksandrs Belovs and Robert Spalek. "Adversary lower bound for the k-sum problem". In: *arXiv preprint arXiv:1206.6528* (2012).

[CN08] Donghoon Chang and Mridul Nandi. "Improved indifferentiability security analysis of chopMD hash function". In: *International Workshop on Fast Software Encryption*. Springer. 2008, pp. 429–443.

[Cor+05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. "Merkle-Damgård revisited: How to construct a hash function". In: *Annual International Cryptology Conference*. Springer. 2005, pp. 430–448.

[CT65] James W Cooley and John W Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of computation* 19.90 (1965), pp. 297–301.

[Cza+18] Jan Czajkowski, Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh. "Post-quantum security of the sponge construction". In: *International Conference on Post-Quantum Cryptography*. Springer. 2018, pp. 185–204.

[Cza+19] Jan Czajkowski, Christian Majenz, Christian Schaffner, and Sebastian Zur. "Quantum Lazy Sampling and Game-Playing Proofs for Quantum Indifferentiability". In: *arXiv preprint arXiv:1904.11477* (2019).

[Dam89] Ivan Bjerre Damgård. "A design principle for hash functions". In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 416–427.

[Dam98] Wim van Dam. "Quantum oracle interrogation: Getting all information for almost half the price". In: *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*. IEEE. 1998, pp. 362–367.

[Din+14] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. "Cryptanalysis of iterated even-mansour schemes with two keys". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2014, pp. 439–457.

[Fey82] Richard P Feynman. "Simulating physics with computers". In: *International journal of theoretical physics* 21.6 (1982), pp. 467–488.

[Gro96] Lov K Grover. "A fast quantum mechanical algorithm for database search". In: *arXiv preprint quant-ph/9605043* (1996).

[HI19] Akinori Hosoyamada and Tetsu Iwata. "Tight Quantum Security Bound of the 4-Round Luby-Rackoff Construction". In: (2019).

[Hos+18] Akinori Hosoyamada, Yu Sasaki, Seiichiro Tani, and Keita Xagawa. "Improved Quantum Multicollision-Finding Algorithm". In: *arXiv preprint arXiv:1811.08097* (2018).

[HST17] Thomas Häner, Damian Steiger, and Matthias Troyer. *ProjectQ*. 2017. URL: https://projectq.ch/.

[JMW17] Stacey Jeffery, Frederic Magniez, and Ronald de Wolf. "Optimal parallel quantum query algorithms". In: *Algorithmica* 79.2 (2017), pp. 509–529.

[LZ18] Qipeng Liu and Mark Zhandry. "On finding quantum multi-collisions". In: *arXiv preprint arXiv:1811.05385* (2018).

[LZ19] Qipeng Liu and Mark Zhandry. "Revisiting Post-Quantum Fiat-Shamir." In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 262.

[Mer89] Ralph C Merkle. "One way hash functions and DES". In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 428–446.

[MRH04] Ueli Maurer, Renato Renner, and Clemens Holenstein. "Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology". In: *Theory of cryptography conference*. Springer. 2004, pp. 21–39.

[Nai+13] Yusuke Naito, Yu Sasaki, Lei Wang, and Kan Yasuda. "Generic state-recovery and forgery attacks on ChopMD-MAC and on NMAC/HMAC". In: *International Workshop on Security*. Springer. 2013, pp. 83–98.

[NC02] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[OR07] David Sena Oliveira and Rubens Viana Ramos. "Quantum bit string comparator: circuits and applications". In: *Quantum Computers and Computing* 7.1 (2007), pp. 17–26.

[Sho94] Peter W Shor. "Algorithms for quantum computation: Discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[Suz+06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. "Birthday paradox for multi-collisions". In: *International Conference on Information Security and Cryptology*. Springer. 2006, pp. 29–40.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. "Post-quantum security of the Fujisaki-Okamoto and OAEP transforms". In: *Theory of Cryptography Conference*. Springer. 2016, pp. 192–216.

[Unr17] Dominique Unruh. "Post-quantum security of Fiat-Shamir". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 65–95.

[Wat18] John Watrous. *The theory of quantum information*. Cambridge University Press, 2018.

[Wol11] Ronald de Wolf. "Quantum computing: Lecture notes". In: *University of Amsterdam* (2011).

[Zha15] Mark Zhandry. *Quantum Query Solvability: A Refinement of Quantum Query Complexity and Applications*. Youtube. 2015. URL: https://www.youtube.com/watch?v=lNCyRE8W2hA&.

[Zha18] Mark Zhandry. "How to Record Quantum Queries, and Applications to Quantum Indifferentiability." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 276.

[Zur19] Sebastian Zur. *Compressed-Fourier-Oracle*. 2019. URL: https://github.com/sebastianzur/Compressed-Fourier-Oracle/.

Appendices

A. Detailed CFO Algorithm

In Algorithm 3 we present the fully-detailed version of Algorithm 1. This algorithm runs the following subroutines:

- Locate, Function 4: This subroutine locates the positions in Δ where the x-entry coincides with the x-entry of the query. The result is represented as q bits, where $q_i = 1 \iff \Delta_i^X = x$. This result is then bitwise XOR'd into an auxiliary register L.
- Add, Function 5: This subroutine adds queried x to the database and takes care of appropriate padding.
- Upd, Function 6: This subroutine updates the database by subtracting η after a suitable basis transformation.
- Rem, Function 7: This subroutine removes $(0, 0)$ entries from the database and puts them to the back in the form of padding.
- Clean, Function 8: This subroutine cleans the auxiliary registers setting them back to initial values.
- Larger, Function 9: This subroutine determines whether one bitstring is larger than a second bitstrings, as described in [OR07].

In the Add and Rem subroutine the unitary P can be found. P permutes the database such that a recently removed entry in the database is moved to the end of the database. Conversely P^{-1} permutes the database such that an empty entry is created in the database as to ensure the correct ordering of the x-entries after adding the query into this newly created empty entry:

$$\mathsf{P}|x_1, ..., x_q\rangle \otimes |y_1, ..., y_n\rangle := |\sigma_n \circ ... \circ \sigma_1(x_1, ..., x_q)\rangle \otimes |y_1, ..., y_n\rangle \,, \tag{A.1}$$

where σ_i is applied conditioned on $y_i = 1$ and is defined by
$\sigma_i(x_1, ..., x_n) := (x_1, ..., x_{i-2}, x_{i-1}, x_{i+1}, x_{i+2}, ..., x_q, x_i)$.

Algorithm 3: Detailed CFO

Input : $|x,\eta\rangle_{XY}|\Delta\rangle_D$
Output: $|x,\eta\rangle_{XY}|\Delta'\rangle_D$

$|a\rangle_A = |0 \in \{0,1\}\rangle_A$ // initialize auxiliary register A
$|l\rangle_L = |0^q \in \{0,1\}^q\rangle_L$ // initialize auxiliary register L
$|l\rangle_L \mapsto \mathsf{Locate}(|x\rangle_X|\Delta\rangle_D|l\rangle_L)$ // locate x in the database
if $l = 0^q$ **then** // if not located
  $|a\rangle_A \mapsto |a \oplus 1\rangle_A$ // save result to register A
if $a = 1$ **then** // if not located
  $|\Delta\rangle_D|l\rangle_L \mapsto \mathsf{Add}(|x\rangle_X|\Delta\rangle_D)$ // add x-entry to the database
$|\Delta^Y\rangle_{D^Y} \mapsto \mathsf{Upd}(|\eta\rangle_Y|\Delta^Y\rangle_{D^Y}|l\rangle_L)$ // update register D^Y
$|\Delta\rangle_D|l\rangle_L \mapsto \mathsf{Rem}(|x\rangle_X|\Delta\rangle_D|l\rangle_L)$ // remove a database entry if $\eta = 0$
$|a\rangle_A \mapsto \mathsf{Clean}(|y\rangle_Y|\Delta^Y\rangle_{D^Y}|l\rangle_L)$ // uncompute register A
$|l\rangle_L \mapsto \mathsf{Locate}(|x\rangle_X|\Delta\rangle_D|l\rangle_L)$ // uncompute register L
return $|x,\eta\rangle_{XY}|\Delta'\rangle_D$ // Δ' is the modified database

Function 4: Locate

Input : $|x\rangle_X|\Delta\rangle_D|l\rangle_L$
Output: $|x\rangle_X|\Delta\rangle_D|l'\rangle_L$

Set $|a\rangle_A = |0 \in \{0,1\}^m\rangle_A$ // initialize auxiliary register A
for $i = 1, ..., q$ **do**
  if $\eta_i \neq 0$ **then** // locate entries in the database
    $|a\rangle_A \mapsto |a \oplus (\Delta_i^X \oplus x)\rangle_A$ // database entry − query
    if $a_i = 0$ **then** // locate matches in the database
      $|l_i\rangle_{L_i} \mapsto |l_i \oplus 1\rangle_{L_i}$ // save the corresponding positions
    $|a\rangle_A \mapsto |a \oplus (\Delta_i^X \oplus x)\rangle_A$ // uncompute register A
return $|x\rangle_X|\Delta\rangle_D|l'\rangle_R$ // l' contains the position of x in Δ

Function 5: Add

Input : $|x\rangle_X|\Delta\rangle_D|l\rangle_L$
Output: $|x\rangle_X|\Delta'\rangle_D|l'\rangle_L$

Set $|a\rangle_A = |0^q \in \{0,1\}^q\rangle_A$ // initialize auxiliary register A
for $i = 1, ..., q$ **do**
  $|a_i\rangle_{A_i} \mapsto \mathsf{Larger}(|\Delta_i^X\rangle_{D_i^X}|x\rangle_X|a_i\rangle_{A_i})$ // check if database entry > query
  if $\Delta_i^Y = 0$ **then** // correct for empty entries
    $|a_i\rangle_{A_i} \mapsto |a_i \oplus 1\rangle_{A_i}$
  for $j = i+1, ..., q$ **do** // flip all higher entries
    $|a_j\rangle_{A_j} \mapsto |a_j \oplus a_i\rangle_{A_j}$ // so we're left with one position
$|\Delta\rangle_D \mapsto \mathsf{P}^{-1}(|\Delta\rangle_D \otimes |a\rangle_A)$ // permute D to create empty entry // P is defined in (A.1)
for $i = 1, ..., q$ **do**
  if $a_i = 1$ **then** // look for this empty entry
    $|\Delta_i^X\rangle_{D_i^X} \mapsto |\Delta_i^X \oplus x\rangle_{D_i^X}$ // add x-entry to the database
    $|l_i\rangle_{L_i} \mapsto |l_i \oplus 1\rangle_{L_i}$ // update location register
if $x \neq 0$ **then** // non zero x implies non zero a
  for $i = 1, ..., q$ **do**
    if $l_i = 1$ **then** // if located
      $|a_i\rangle_{A_i} \mapsto |a_i \oplus 1\rangle_{A_i}$ // uncompute register A
return $|x\rangle_X|\Delta'\rangle_D|l'\rangle_L$ // Δ' is the modified database // l' is modified l

Function 6: Upd

Input : $|\eta\rangle_Y|\Delta^Y\rangle_{D^Y}|l\rangle_L$
Output: $|\eta\rangle_Y|\Delta'^Y\rangle_{D^Y}|l\rangle_L$

for $i = 1, ..., q$ **do**
  if $l_i = 1$ **then** // if located
    $|\Delta_i^Y\rangle_{D_i^Y} \mapsto |\Delta_i^Y \oplus \eta\rangle_{D_i^Y}$ // update the Y register of entry
return $|\eta\rangle_Y|\Delta'^Y\rangle_{D^Y}|l\rangle_L$ // Δ'^Y is modified Y register of the database

Function 7: Rem

Input : $|x\rangle_X|\Delta\rangle_D|l\rangle_L$
Output: $|x\rangle_X|\Delta'\rangle_D|l'\rangle_L$

Set $|a\rangle_A = |0^q \in \{0,1\}^q\rangle_A$ // initialize auxiliary register A
Set $|b\rangle_B = |0 \in \{0,1\}\rangle_B$ // initialize auxiliary register B
for $i = 1, ..., q$ **do**
  if $l_i = 1$ **then**
    if $\eta_i = 0$ **then** // if entry is incorrect
      $|\Delta_i^X\rangle_{D_i^X} \mapsto |\Delta_i^X \oplus x\rangle_{D_i^X}$ // remove the entry
      $|b\rangle_B \mapsto |b \oplus 1\rangle_B$ // save that we have removed an entry
if $b = 1$ **then** // if we removed an entry
  for $i = 1, ..., q$ **do**
    $|a_i\rangle_{A_i} \mapsto \mathsf{Larger}(|x\rangle_X, |\Delta_i^X\rangle_{D_i^X}, |a_i\rangle_{A_i})$ // check if query > database entry
    if $\Delta_i^Y = 0$ **then** // correct for empty entries
      $|a_i\rangle_{A_i} \mapsto |a_i \oplus 1\rangle_{A_i}$
    for $j = i - 1, ..., 1$ **do** // flip all lower entries
      $|a_j\rangle_{A_j} \mapsto |a_j \oplus a_i\rangle_{A_j}$ // so we're left with only the removed position
    $|l_i\rangle_{L_i} \mapsto |l_i \oplus a_i\rangle_{L_i}$ // correct for the removed entry
  $|\Delta\rangle_D \mapsto P(|\Delta\rangle_D \otimes |a\rangle_A)$ // permute D to move the empty entry
  for $i = q, ..., 1$ **do** // uncompute register A
    for $j = q, ..., i + 1$ **do** // by calculating the first position
      $|a_j\rangle_{A_j} \mapsto |a_j \oplus a_i\rangle_{A_j}$ // such that database entry > query
    if $\Delta_i^Y \neq 0$ **then** // as in the Add subroutine
      $|a_i\rangle_{A_i} \mapsto |a_i \oplus 1\rangle_{A_i}$
    $|a_i\rangle_{A_i} \mapsto \mathsf{Larger}(|\Delta_i^X\rangle_{D_i^X}, |x\rangle_X, |a_i\rangle_{A_i})$
$|a\rangle_A \mapsto \mathsf{Locate}(|x\rangle_X|\Delta\rangle_D|l\rangle_A)$
if $a \neq 0^q$ **then** // check if we have removed
  $|b\rangle_B \mapsto |b \oplus 1\rangle_B$ // uncompute register B
$|a\rangle_A \mapsto \mathsf{Locate}(|x\rangle_X|\Delta\rangle_D|l\rangle_A)$ // uncompute register A
return $|x\rangle_X|\Delta'\rangle_D|l'\rangle_L$ // Δ' is modified database
// l' is modified l

Function 8: Clean

Input : $|\eta\rangle_Y|\Delta^Y\rangle_D|l\rangle_L|a\rangle_A$
Output: $|\eta\rangle_Y|\Delta^Y\rangle_D|l\rangle_L|a'\rangle_A$

Set $|b\rangle_B = |0 \in \{0,1\}^n\rangle_B$ // initialize auxiliary register B
for $i = 1, ..., q$ **do**
  if $l_i = 1$ **then**
    $|b\rangle_B \mapsto |b + (\Delta_i^Y - \eta)\rangle_B$ // database entry − query
    if $b = 0$ **then** // locate matches in the database
      if $\eta \neq 0$ **then** // if we added
        $|a\rangle_A \to |a \oplus 1\rangle_A$
    $|b\rangle_B \mapsto |b - (\Delta_i^Y - \eta)\rangle_B$ // uncompute register B
return $|\eta\rangle_Y|\Delta^Y\rangle_D|l\rangle_L|a'\rangle_A$ // a' is modified register A

Function 9: Larger

Input : $|u\rangle_U, |v\rangle_V, |r\rangle_R$
Output: $|u\rangle_U, |v\rangle_V, |r'\rangle_R$

Set $|a\rangle_A = |0^{3t-1} \in \{0,1\}^{3t-1}\rangle_A$ // initialize auxiliary register A
for $i = 1, .., t-1$ **do** // t denotes length of bit strings u, v
  $|a_{3i-2}\rangle_{A_{3i-2}} \mapsto |a_{3i-2} \oplus u_i(1-v_i)\rangle_{A_{3i-2}}$ // save if $u_i > v_i$
  $|a_{3i-1}\rangle_{A_{3i-1}} \mapsto |a_{3i-1} \oplus v_i(1-u_i)\rangle_{A_{3i-1}}$ // save if $v_i > u_i$
  $|a_{3i}\rangle_{A_{3i}} \mapsto |a_{3i} \oplus (1-a_{3i})(1-a_{3i})\rangle_{A_{3i}}$ // save if they're equal
$|a_{3t-2}\rangle_{A_{3t-2}} \mapsto |a_{3t-2} \oplus u_t(1-v_t)\rangle_{A_{3t-2}}$ // save if $u_t > v_t$
$|a_{3t-1}\rangle_{A_{3t-1}} \mapsto |a_{3t-1} \oplus v_t(1-u_t)\rangle_{A_{3t-1}}$ // save if $v_t > u_t$
for $i = t-1, .., 1$ **do** // the first bit difference is dominant
  $|a_{3i-2}\rangle_{A_{3i-2}} \mapsto |a_{3i-2} \oplus (1-a_{3i})(1-a_{3i+1})\rangle_{A_{3i-2}}$
  $|a_{3i-1}\rangle_{A_{3i-1}} \mapsto |a_{3i-1} \oplus (1-a_{3i})(1-a_{3i+2})\rangle_{A_{3i-1}}$
  // however if equal, then next bit becomes dominant
$|r\rangle_R \mapsto |z \oplus a_1\rangle$ // $a_1 = 1$ if $u > v$
for $i = 1, .., t-1$ **do** // uncompute register A by repeating above
  $|a_{3i-1}\rangle_{A_{3i-1}} \mapsto |a_{3i-1} \oplus (1-a_{3i})(1-a_{3i+2})\rangle_{A_{3i-1}}$ // calculation in
  $|a_{3i-2}\rangle_{A_{3i-2}} \mapsto |a_{3i-2} \oplus (1-a_{3i})(1-a_{3i+1})\rangle_{A_{3i-2}}$ // reverse order
$|a_{3t-1}\rangle_{A_{3t-1}} \mapsto |a_{3t-1} \oplus v_t(1-u_t)\rangle_{A_{3t-1}}$
$|a_{3t-2}\rangle_{A_{3t-2}} \mapsto |a_{3t-2} \oplus u_t(1-v_t)\rangle_{A_{3t-2}}$
for $i = t-1, .., 1$ **do**
  $|a_{3i}\rangle_{A_{3i}} \mapsto |a_{3i} \oplus (1-a_{3i})(1-a_{3i})\rangle_{A_{3i}}$
  $|a_{3i-1}\rangle_{A_{3i-1}} \mapsto |a_{3i-1} \oplus v_i(1-u_i)\rangle_{A_{3i-1}}$
  $|a_{3i-2}\rangle_{A_{3i-2}} \mapsto |a_{3i-2} \oplus u_i(1-v_i)\rangle_{A_{3i-2}}$
return $|r'\rangle_R$ // r' is modified r

B. Explicit construction of $P_{r,k}$

In this appendix we give an explicit construction of the projector $P_{r,k}$ for $\mathsf{R}^r_{\mathsf{Col}}$, but the same approach works for any relation. For $i_{1,1}, ..., i_{k,r} \in [q]$ and $y_1, ..., y_k \in \{0,1\}^n$ define

$$P_{i_{1,1},...,i_{k,r},y_1,...,y_k} = |y_1\rangle\langle y_1|_{D^Y_{i_{1,1}}} \otimes \cdots \otimes |y_k\rangle\langle y_k|_{D^Y_{k,r}}\,, \tag{B.1}$$

which projects onto the database satisfying $D^Y_{i_{1,1}} = \cdots = D^Y_{i_{1,r}} = y_1$, ... ,
$D^Y_{i_{k,1}} = \cdots = D^Y_{i_{k,r}} = y_k$. This gives a specific instance of the database containing k distinct $r-$tuples satisfying $\mathsf{R}^r_{\mathsf{Col}}$, but by varying over $i_{1,1}, ..., i_{k,r}, y_1, ..., y_k$ we include every possible case. Let $\left\{P_{i_{1,1},...,i_{k,r},y_1,...,y_k}\right\}$ be the set of all these projectors and denote its cardinality by $|P|$. We can refer to the elements in $\left\{P_{i_{1,1},...,i_{k,r},y_1,...,y_k}\right\}$ by $P_1, ..., P_{|P|}$ (the ordering does not matter) to create $P_{r,k}$ by the following Gram-Schmidt-like process:

$$P_{r,k} = P_1 + P_2(\mathbb{I} - P_1) + \cdots + P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1). \tag{B.2}$$

As every term in the sum $P_1 + P_2(\mathbb{I} - P_1) + \cdots + P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1)$ is orthogonal to each previous term, we see that

$$\begin{aligned} P^2_{r,k} &= \left(P_1 + P_2(\mathbb{I} - P_1) + \cdots + P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1)\right)^2 \\ &= P_1^2 + (P_2(\mathbb{I} - P_1))^2 + \cdots + (P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1))^2 \\ &= P_1 + P_2(\mathbb{I} - P_1) + \cdots + P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1) \\ &= P_{r,k}. \end{aligned} \tag{B.3}$$

To proof correctness of this construction there remains to show that
$\text{span}\left\{P_1, ..., P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1)\right\} = \text{span}\left\{P_1, ..., P_{|P|}\right\}$. We do this by induction on $|P|$. Since the claim is trivial for $|P| = 1$, let us assume that
$\text{span}\left\{P_1, ..., P_{|P|-1}(\mathbb{I} - P_{|P|-2}) \cdots (\mathbb{I} - P_1)\right\} = \text{span}\left\{P_1, ..., P_{|P|-1}\right\}$ and choose
$x \in \text{span}\left\{P_1, ..., P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1)\right\}$.

$$\begin{aligned} x \in \text{span}&\left\{P_1, ..., , P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1)\right\} \\ &\iff x \in \text{span}\left\{P_1, ..., P_{|P|-1}(\mathbb{I} - P_{|P|-2}) \cdots (\mathbb{I} - P_1)\right\} \\ &\qquad \vee\, x \in \text{span}\left\{P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1)\right\} \\ &\iff x \in \text{span}\left\{P_1, ..., P_{|P|-1}\right\} \\ &\qquad \vee\, x \in \text{span}\left\{P_{|P|}(\mathbb{I} - P_{|P|-1}) \cdots (\mathbb{I} - P_1)\right\} \\ &\iff x \in \text{span}\left\{P_1, ..., P_{|P|-1}\right\} \vee x \in \text{span}\left\{P_{|P|}\right\} \\ &\iff x \in \text{span}\left\{P_1, ..., P_{|P|}\right\}. \end{aligned} \tag{B.4}$$

C. Proof of Lemma 3.4.2

Lemma 3.4.2. *Let the series $a_{q,n}, b_{q,n}$ be defined as follows:*

- $a_{q,0} := q$,
- $b_{q,0} := q$,
- $a_{q,n} := \sum_{i=1}^{n} \sum_{j=0}^{q-1} c^i a_{j,n-i}$,
- $b_{q,n} := \sum_{i=1}^{n} \sum_{j=0}^{q-1} (c^i b_{j,n-i} + d)$,

for constants c, d. Then $\forall n \geq 1$ it holds that

$$\begin{aligned} a_{q,n} &= \frac{q^n}{n!} 2^{n-1} c^n, \\ b_{q,n} &= a_{q,n} + \sum_{i=0}^{n-1} (n-i) a_{q,i} d. \end{aligned} \tag{C.1}$$

Proof. The lemma clearly holds for $n = 1$. Assume both statements hold up to $n-1$. For the first series we simply have

$$\begin{aligned} a_{q,n} &= \sum_{i=1}^{n} \sum_{j=0}^{q-1} c^i a_{j,n-i} \\ &= \frac{q^n}{n!} c^n + \sum_{i=1}^{n-1} \sum_{j=0}^{q-1} \frac{j^n}{n!} c^i 2^{n-i-1} c^{n-i} \\ &= \frac{q^n}{n!} c^n \left(1 + \sum_{i=0}^{n-2} 2^i \right) = \frac{q^n}{n!} 2^{n-1} c^n. \end{aligned} \tag{C.2}$$

For the second series, we start with

$$\begin{aligned} b_{q,n} &= \sum_{i=1}^{n} \sum_{j=0}^{q-1} \left(c^i b_{j,n-i} + d \right) \\ &= \frac{q^n}{n!} c^n + \frac{q^n}{n!} nd + \sum_{i=1}^{n-1} \sum_{j=0}^{q-1} c^i b_{j,n-i} \\ &= \frac{q^n}{n!} c^n + \frac{q^n}{n!} nd + \sum_{i=1}^{n-1} \sum_{j=0}^{q-1} \left(a_{j,n-i} + \sum_{k=0}^{n-i-1} (n-i-k) a_{j,k} d \right). \end{aligned} \tag{C.3}$$

By noting that $\frac{q^n}{n!}c^n + \sum_{i=1}^{n-1}\sum_{j=0}^{q-1} c^i a_{j,n-i} = \sum_{i=1}^{n}\sum_{j=0}^{q-1} c^i a_{j,n-i} = a_{q,n}$ we find

$$\begin{aligned}
&\frac{q^n}{n!}c^n + \frac{q^n}{n!}nd + \sum_{i=1}^{n-1}\sum_{j=0}^{q-1} c^i \left(a_{j,n-i} + \sum_{k=0}^{n-i-1} (n-i-k)a_{j,k}d \right) \\
&= a_{q,n} + \frac{q^n}{n!}nd + \sum_{i=1}^{n-1}\sum_{j=0}^{q-1}\sum_{k=0}^{n-i-1} (n-i-k)c^i a_{j,k}d \\
&= a_{q,n} + \frac{q^n}{n!}nd + \sum_{i=1}^{n-1} \frac{q^n}{n!}(n-i)c^i d + \sum_{i=1}^{n-1}\sum_{k=1}^{n-i-1} \frac{q^n}{n!}(n-i-k)2^{n-1}c^n d \\
&= a_{q,n} + \frac{q^n}{n!}nd + \sum_{l=1}^{n-1} \frac{q^n}{n!}(n-l)c^l d + \sum_{i=1}^{n-1}\sum_{l=i+1}^{n-1} \frac{q^n}{n!}(n-l)2^{l-i-1}c^l d, \qquad \text{(C.4)}
\end{aligned}$$

where in the second equality we have used that for $n \geq 1$ we just proved that $a_{q,n} = \frac{q^n}{n!}2^{n-1}c^n$.

$$\begin{aligned}
&a_{q,n} + \frac{q^n}{n!}\left(nd + \sum_{l=1}^{n-1}(n-l)c^l d + \sum_{i=1}^{n-1}\sum_{l=i+1}^{n-1}(n-l)2^{l-i-1}c^l d \right) \\
&= a_{q,n} + \frac{q^n}{n!}\left(nd + \sum_{l=1}^{n-1}(n-l)c^l d\left(1 + \sum_{i=1}^{n-1} 2^{l-i-1} \right) \right) \\
&= a_{q,n} + \frac{q^n}{n!}\left(nd + \sum_{l=1}^{n-1}(n-l)c^l d\left(1 + \sum_{i=0}^{l-2} 2^{i} \right) \right) \\
&= a_{q,n} + \frac{q^n}{n!}\left(nd + \sum_{l=1}^{n-1}(n-l)2^{l-1}c^l d \right) \\
&= a_{q,n} + \sum_{l=0}^{n-1}(n-l)a_{q,l}d. \qquad \text{(C.5)}
\end{aligned}$$

□

D. The $r = 4$ case for Theorem 4.1.1

Inserting our bound for $\|P_{3,k}|\psi_q\rangle\|$ into Equation 4.12 yields

$$\|P_{4,k}|\psi_q\rangle\| \leq \left(\frac{eq\sqrt{M_3}}{k\sqrt{N}}\right)^k + e^{q\sqrt{\frac{M_3}{N}}} q \left(\left(\frac{\sqrt{2}e^{3/2}q^{7/4}}{M_3 N^{3/4}} + \mathcal{O}\left(\left(\frac{M_3}{N}\right)^{1/48}\right)\right)^{M_3} + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right)\right). \tag{D.1}$$

Once again we need $\frac{\sqrt{2}e^{3/2}q^{7/4}}{N^{3/4}} < M_2 < \frac{kN}{(eq)^2}$ for both $\left(\frac{eq\sqrt{M_3}}{k\sqrt{N}}\right)^k$ and $\frac{\sqrt{2}e^{3/2}q^{7/4}}{M_3 N^{3/4}} + \mathcal{O}\left(\left(\frac{M_3}{N}\right)^{1/48}\right)$ to be smaller than 1. In the case of $q > \sqrt{kN}$ this results in $k > \frac{\sqrt{2}e^{3/2}q^{7/4}}{N^{3/4}} > k^{7/8}N^{1/8}$ and thus $k > N$. As this implies $q > N$, we conclude that in this case the bound becomes trivial, i.e. 1.

We choose $M_3 = \frac{(2e)^{3/2}q^{7/4}}{N^{3/4}}$, where we can bound $q\sqrt{\frac{M_3}{N}} \leq 2ek^{15/16}N^{1/16}$ in the case of $q \leq \sqrt{kN}$.

$$\begin{aligned}
&e^{q\sqrt{\frac{M_3}{N}}} q \left(\left(\frac{\sqrt{2}e^{3/2}q^{7/4}}{M_3 N^{3/4}} + \mathcal{O}\left(\left(\frac{M_3}{N}\right)^{1/48}\right)\right)^{M_3} + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right)\right) \\
&\leq e^{2ek^{15/16}N^{1/16}} N \left(\left(\frac{1}{2} + \mathcal{O}\left(\left(\frac{M_3}{N}\right)^{1/48}\right)\right)^{M_3} + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right)\right) \\
&\leq \left(\frac{1}{2}\right)^{-8k^{15/16}N^{1/16}} \mathcal{O}\left(2^{N^{1/16}}\right) \left(\mathcal{O}\left(\frac{1}{2}\right)^{M_3} + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right)\right),
\end{aligned} \tag{D.2}$$

where in the last inequality we used that $M_3 < k \leq N$. Again we update $M_3 = \max\left\{\frac{(2e)^{3/2}q^{7/4}}{N^{3/4}}, 10k^{15/16}N^{1/16}\right\}$ such that we obtain

$$\begin{aligned}
&\left(\frac{1}{2}\right)^{-8k^{15/16}N^{1/16}} \mathcal{O}\left(2^{N^{1/16}}\right) \left(\mathcal{O}\left(\frac{1}{2}\right)^{M_3} + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right)\right) \\
&\leq \left(\frac{1}{2}\right)^{-8k^{15/16}N^{1/16}} \mathcal{O}\left(2^{N^{1/16}}\right) \left(\mathcal{O}\left(\frac{1}{2}\right)^{10k^{15/16}N^{1/16}} + \mathcal{O}\left(2^{-k^{7/8}N^{1/8}}\right)\right) \\
&\leq \mathcal{O}\left(2^{-k^{15/16}N^{1/16}}\right).
\end{aligned} \tag{D.3}$$

Like before, by noting that $\frac{(2e)^{3/2}q^{7/4}}{N^{3/4}} \leq 10N^{15/16}N^{1/16}$ implies $q \leq \frac{10^{4/7}k^{15/28}N^{13/28}}{(2e)^{6/7}}$, the final bound becomes

$$\begin{aligned}\|P_{4,k}|\psi_q\rangle\| &\leq \left(\frac{eq\sqrt{\frac{(2e)^{3/2}q^{7/4}}{N^{3/4}}}}{k\sqrt{N}} + \frac{e\frac{10^{4/7}k^{15/28}N^{13/28}}{(2e)^{6/7}}\sqrt{10k^{15/16}N^{1/16}}}{k\sqrt{N}}\right)^k + \mathcal{O}\left(2^{-k^{15/16}N^{1/16}}\right) \\ &\leq \left(\frac{2^{3/4}e^{7/4}q^{15/8}}{kN^{7/8}} + \mathcal{O}\left(\left(\frac{k}{N}\right)^{1/224}\right)\right)^k + \mathcal{O}\left(2^{-k^{15/16}N^{1/16}}\right) \qquad \text{(D.4)}\end{aligned}$$

and it still holds that

$$\begin{aligned}q\sqrt{\frac{M_3}{N}} &\leq \frac{10^{15/14}k^{225/224}}{(2e)^{6/7}N^{1/224}} \\ &\leq \frac{10^{15/14}k^{15/16}N^{1/16}}{(2e)^{6/7}} \\ &\leq 2ek^{15/16}N^{1/16}. \qquad \text{(D.5)}\end{aligned}$$

E. The inductive step for Theorem 4.3.1

Theorem 4.3.1. *Let $P_{r,k}$ be the projector which projects onto the database containing at least k distinct r-tuples satisfying $\mathsf{R}^r_{\mathsf{Col}}$, as defined in Definition 3.1.1. Also define $\gamma'(r) = 2^{(2^{r-2}-1)/2^{r-2}} e^{3(2^{r-1}-1)/2^{r-1}} r!$. Then for any $M_{r-1} \leq N$, $k \leq N$, constant $r \geq 2$ and parallel query size p it holds that*

$$\|P_{r,k}|\psi_q\rangle\| \leq \left(\frac{\gamma'(r) q^{(2^r-1)/2^{r-1}} p}{k N^{(2^{r-1}-1)/2^{r-1}}} + \mathcal{O}\left(\left(\frac{k}{N} \right)^{1/(2^{r-1}-1)2^{r+1}} \right) \right)^k + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r} N^{1/2^r}} \right). \tag{E.1}$$

Assume the theorem holds for up to r. Then by Equation 4.34 we know that

$$\begin{aligned}
&\|P_{r+1,k}|\psi_q\rangle\| \\
&\leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} \sum_{t_1,\ldots,t_j=1}^{r+1} \left(\sum_{l_1,\ldots,l_r=0}^{N} \frac{l_{r-t_1}(ep)^{t_1}}{(t_1 N)^{t_1}} \cdots \frac{r - l_{t_j}(ep)^{t_j}}{(t_j N)^{t_j}} \right. \\
&\quad \left. \cdot \|\hat{P}_{1,l_1} \cdots \hat{P}_{r,l_r} P_{r+1,k-j}(\mathbb{I} - P_{r+1,k})|\psi_i\rangle\|^2 \right)^{1/2} \\
&\leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} \sum_{\#(t=1)=0}^{j} \cdots \sum_{\#(t=r)=0}^{j-\cdots-\#(t=r-1)} \binom{j}{\#(t=1)} \cdots \binom{j-\cdots-\#(t=r-1)}{\#(t=r)} \\
&\quad \cdot \left(\sum_{l_1,\ldots,l_r=0}^{N} \left(\frac{l_r ep}{N} \right)^{\#(t=1)} \cdots \left(\frac{N(ep)^{r+1}}{((r+1)N)^{r+1}} \right)^{j-\cdots-\#(t=r-1)} \right. \\
&\quad \left. \cdot \|\hat{P}_{1,l_1} \cdots \hat{P}_{r,l_r} P_{r+1,k-j}(\mathbb{I} - P_{r+1,k})|\psi_i\rangle\|^2 \right)^{1/2}.
\end{aligned} \tag{E.2}$$

By applying the same trick as in Equation 4.38 after introducing $M_1, ..., M_r$ we find

$$
\begin{aligned}
&\|P_{r+1,k}|\psi_q\rangle\| \\
&\leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} \sum_{\#(t=1)=0}^{j} \cdots \sum_{\#(t=r)=0}^{j-\cdots-\#(t=r-1)} \binom{j}{\#(t=1)} \cdots \binom{j-\cdots-\#(t=r-1)}{\#(t=r)} \\
&\quad \cdot \Bigg(\sum_{l_1,\ldots,l_r=0}^{N} \left(\frac{l_r ep}{N}\right)^{\#(t=1)} \cdots \left(\frac{N(ep)^{r+1}}{((r+1)N)^{r+1}}\right)^{j-\cdots-\#(t=r-1)} \\
&\quad \cdot \|\hat{P}_{1,l_1} \cdots \hat{P}_{r,l_r} P_{r+1,k-j}(\mathbb{I} - P_{r+1,k})|\psi_i\rangle\|^2 \Bigg)^{1/2} \\
&\leq \sum_{i=0}^{q-1} \sum_{j=1}^{k} \Bigg(\sum_{\#(t=1)=0}^{j} \cdots \sum_{\#(t=r)=0}^{j-\cdots-\#(t=r-1)} \binom{j}{\#(t=1)} \cdots \binom{j-\cdots-\#(t=r-1)}{\#(t=r)} \\
&\quad \cdot \Bigg(\left(\frac{M_r ep}{N}\right)^{\#(t=1)} \cdots \left(\frac{M_r (ep)^{r-1}}{((r-1)N)^{r-1}}\right)^{\#(t=r-1)} \left(\frac{N(ep)^r}{(rN)^r}\right)^{j-\cdots-\#(t=r-1)} \\
&\quad \cdot \|P_{r,k-j}(\mathbb{I} - P_{r,k})|\psi_i\rangle\|^2 \Bigg)^{1/2} + \|P_{r-1,M_{r-1}+1}|\psi_i\rangle\| + \cdots + \|P_{1,M_1+1}|\psi_i\rangle\| \Bigg). \qquad \text{(E.3)}
\end{aligned}
$$

As before we have to set $M_1 = qp$ and based on our bounds of $\|P_{r,k}|\psi_q\rangle\|$ for $r = 2, ..., r$ we choose $M_r = \max\left\{2\gamma'(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}}, (4r!+2)k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}\right\}$. For all the respective left and right values in this maximum it holds that $\frac{M_r ep}{N} > \cdots > \frac{N(ep)^{r+1}}{((r+1)N)^{r+1}}$ and thus the inequalities also hold for the maximum of both values. By bounding $(4r! + 2) \leq 5r!$ we see that $2\gamma'(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}} \leq (4r! + 2)k^{(2^{r+1}-1)/2^{r+1}}N^{1/2^{r+1}}$ implies $q \leq \frac{5^{2^{r-1}/(2^r-1)}ek^{(2(2^r-1)+1)/4(2^r-1)}N^{(2(2^r-1)-1)/4(2^r-1)}}{(2)^{(2^r-2)/(2^r-1)}p^{2^{r-1}/(2^r-1)}}$.

As such we can simplify

$$
\begin{aligned}
&\sum_{i=0}^{q-1}\sum_{j=1}^{k}\Bigg(\sum_{\#(t=1)=0}^{j}\cdots\sum_{\#(t=r)=0}^{j-\cdots-\#(t=r-1)}\binom{j}{\#(t=1)}\cdots\binom{j-\cdots-\#(t=r-1)}{\#(t=r)}\\
&\quad\cdot\Bigg(\left(\frac{M_r ep}{N}\right)^{\#(t=1)}\cdots\left(\frac{M_r(ep)^{r-1}}{((r-1)N)^{r-1}}\right)^{\#(t=r-1)}\left(\frac{N(ep)^r}{(rN)^r}\right)^{j-\cdots-\#(t=r-1)}\\
&\quad\cdot\|P_{r,k-j}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|^2\Bigg)^{1/2}+\|P_{r-1,M_{r-1}+1}|\psi_i\rangle\|+\cdots+\|P_{1,M_1+1}|\psi_i\rangle\|\Bigg)\\
&\leq\sum_{i=0}^{q-1}\sum_{j=1}^{k}\Bigg(\sum_{\#(t=1)=0}^{j}\cdots\sum_{\#(t=r)=0}^{j-\cdots-\#(t=r-1)}\binom{j}{\#(t=1)}\cdots\binom{j-\cdots-\#(t=r-1)}{\#(t=r)}\\
&\quad\cdot\sqrt{\left(\frac{M_r ep}{N}\right)^{j}}\|P_{r,k-j}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|+\|P_{r-1,M_{r-1}+1}|\psi_i\rangle\|+\cdots+\|P_{2,M_2+1}|\psi_i\rangle\|\Bigg)\\
&=\sum_{i=0}^{q-1}\sum_{j=1}^{k}\Bigg((r+1)^j\sqrt{\left(\frac{M_r ep}{N}\right)^{j}}\|P_{r,k-j}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|\\
&\quad+\|P_{r-1,M_{r-1}+1}|\psi_i\rangle\|+\cdots+\|P_{2,M_2+1}|\psi_i\rangle\|\Bigg)\\
&\leq\sum_{i=0}^{q-1}\sum_{j=1}^{k}\Bigg((r+1)^j\sqrt{\left(\frac{M_r ep}{N}\right)^{j}}\|P_{r,k-j}(\mathbb{I}-P_{r,k})|\psi_i\rangle\|\\
&\quad+\|P_{r-1,M_{r-1}+1}|\psi_q\rangle\|+\cdots+\|P_{2,M_2+1}|\psi_q\rangle\|\Bigg).
\end{aligned}
\tag{E.4}
$$

By using Lemma 3.4.2 we arrive at

$$
\begin{aligned}
&\sum_{i=0}^{q-1}\sum_{j=1}^{k}\Bigg((r+1)^j\sqrt{\left(\frac{M_r ep}{N}\right)^j}\,\|P_{r,k-j}(\mathbb{1}-P_{r,k})|\psi_i\rangle\| \\
&\qquad +\|P_{r-1,M_{r-1}+1}|\psi_q\rangle\|+\cdots+\|P_{2,M_2+1}|\psi_q\rangle\|\Bigg) \\
&=\frac{q^k}{k!}\left(\frac{(r+1)\sqrt{M_r ep}}{\sqrt{N}}\right)^k+\sum_{j=1}^{k}\frac{q^j}{j!}\left(\frac{(r+1)\sqrt{M_r ep}}{\sqrt{N}}\right)^{j-1} \\
&\quad(\|P_{r,M_r+1}|\psi_q\rangle\|+\cdots+\|P_{2,M_2+1}|\psi_q\rangle\|) \\
&\leq\left(\frac{(r+1)eq\sqrt{M_r ep}}{k\sqrt{N}}\right)^k+\sum_{j=1}^{k}\frac{q^j}{j!}\left(\frac{(r+1)\sqrt{M_r ep}}{\sqrt{N}}\right)^{j-1} \\
&\quad(\|P_{r,M_r+1}|\psi_q\rangle\|+\cdots+\|P_{2,M_2+1}|\psi_q\rangle\|) \\
&\leq\left(\frac{(r+1)eq\sqrt{M_r ep}}{k\sqrt{N}}\right)^k+e^{(r+1)q\sqrt{\frac{M_r ep}{N}}}q\left(\|P_{r,M_r+1}|\psi_q\rangle\|+\cdots+\|P_{2,M_2+1}|\psi_q\rangle\|\right). \qquad \text{(E.5)}
\end{aligned}
$$

For this bound to be non-trivial in the case of $q>\sqrt{\frac{kN}{p}}$, we need $\gamma'(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}}<M_r<\frac{k}{e^2}<k$, which results in $k>k^{(2^r-1)/2^r}(Np)^{1/2^{r-1}}$ which implies $k>Np$ and thus results $q>\frac{N}{p}$. As this contradicts the fact that $qp\leq N$ we arrive at a trivial bound, i.e. 1.

In the case that $q \leq \sqrt{\frac{kN}{p}}$ we can bound the second term as

$$
\begin{aligned}
& e^{(r+1)q\sqrt{\frac{M_r ep}{N}}} q \left(\left(\frac{\gamma'(r) q^{(2^r-1)/2^{r-1}} p}{M_r N^{(2^{r-1}-1)/2^{r-1}}} + \mathcal{O}\left(\left(\frac{M_r}{N} \right)^{1/(2^{r-1}-1)2^{r+1}} \right) \right)^{M_r} \right. \\
& \quad \left. + \cdots + \mathcal{O}\left(2^{-k^{(2^r-1)/2^r} N^{1/2^r}} \right) \right) \\
& \leq e^{(r+1)! e k^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} N \left(\left(\frac{1}{2} \right. \right. \\
& \quad \left. \left. + \mathcal{O}\left(\left(\frac{k}{N} \right)^{(2^{r+1}-1)/2^{2(r+1)}(2^{r-1}-1)} \right) \right)^{(4(r+1)!+2)k^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} \right. \\
& \quad \left. + \cdots + \left(\frac{1}{2} \right)^{26k^{7/8}N^{1/8}} \right) \\
& \leq \mathcal{O}\left(\frac{1}{2} \right)^{-4(r+1)! e k^{(2^{r+1}-1)/2^{r+1}}} \mathcal{O}\left(2^{N^{1/2^{r+1}}} \right) \mathcal{O}\left(\frac{1}{2} \right)^{(4(r+1)!+2)k^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} \\
& \leq \mathcal{O}\left(2^{-k^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} \right). \qquad \text{(E.6)}
\end{aligned}
$$

Combining our results yields

$$
\begin{aligned}
& \| P_{r+1,k} |\psi_q\rangle \| \\
& \leq \left(\frac{eq\sqrt{2\gamma'(r)\frac{q^{(2^r-1)/2^{r-1}}}{N^{(2^{r-1}-1)/2^{r-1}}}} ep}{k\sqrt{N}} \right. \\
& \quad \left. + \frac{e \frac{5^{2^{r-1}/(2^r-1)} e k^{(2(2^r-1)+1)/4(2^r-1)} N^{(2(2^r-1)-1)/4(2^r-1)}}{(2)^{(2^r-2)/(2^r-1)} p^{2^{r-1}/(2^r-1)}} \sqrt{(4r!+2)k^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} ep}{k\sqrt{N}} \right)^k \\
& \quad + \mathcal{O}\left(2^{-k^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} \right) \\
& \leq \left(\frac{\gamma'(r+1) q^{(2^{r+1}-1)/2^r} p}{k N^{(2^r-1)/2^r}} + \mathcal{O}\left(\left(\frac{k}{N} \right)^{1/(2^r-1)2^{r+2}} \right) \right)^k + \mathcal{O}\left(2^{-k^{(2^{r+1}-1)/2^{r+1}} N^{1/2^{r+1}}} \right), \qquad \text{(E.7)}
\end{aligned}
$$

which concludes the induction.