

Université de Paris

Thèse de doctorat en Informatique

Quantum Algorithms for the Monte Carlo Method

Yassine HAMOUDI

Présentée et soutenue publiquement le 7 juillet 2021.

Directeur de thèse :	Frédéric MAGNIEZ	Directeur de recherche (Université de Paris, CNRS)
Co-directeur de thèse :	Miklos SANTHA	Directeur de recherche (Université de Paris, CNRS)
Rapporteurs :	Ashley MONTANARO	Professeur (University of Bristol)
	Michele MOSCA	Professeur (University of Waterloo)
Examineurs :	Omar FAWZI	Directeur de recherche (ENS de Lyon, Inria)
	Stacey JEFFERY	Docteur (Centrum Wiskunde & Informatica)
	María NAYA-PLASENCIA	Directrice de recherche (Inria de Paris)

Abstract

The Monte Carlo method is a central paradigm of algorithm design that consists of using statistical analysis and random sampling to solve problems that have a probabilistic interpretation. This thesis explores the advantages offered by a quantum computer to speed up this method.

The first part of our work concerns the problem of estimating average values in a time-efficient way. We develop new quantum algorithms for estimating the mean of a real-valued random variable obtained as the output of a quantum computation. Our estimators achieve a near-optimal quadratic speedup over the number of classical samples required to get a sub-Gaussian error rate or an (ϵ, δ) -approximation guarantee. Furthermore, we describe a framework that provides a notion of “stopping time” for a quantum process generating a random variable. We show that the mean estimation problem can be further sped up when the average stopping time of the underlying process is small. These techniques are applied to the construction of a near-optimal quantum query algorithm for approximately counting the number of triangles in a graph.

In the second part, we study the task of importance sampling and its applications to stochastic optimization. We construct a quantum algorithm for sampling multiple elements from a finite distribution specified as a probability vector. Our approach solves the more general problem of preparing multiple copies of a quantum state whose amplitudes are given by a query oracle. We illustrate the use of this result by constructing two hybrid quantum-classical algorithms based on the multiplicative weight update and stochastic gradient descent methods. Our two applications address the problems of online prediction with expert advice and minimizing a submodular set function.

In the third part, we consider quantum algorithms that operate with limited memory. We first study the problem of approximating the frequency moments in the data stream model with multiple passes over the input. We construct a quantum algorithm that uses less memory than the best possible classical streaming algorithms. Our method combines the above-mentioned quantum mean estimators with reversible simulation techniques. Then, we examine the limitations of space-bounded algorithms in the quantum query model. We develop a new approach for proving time-space tradeoff lower bounds in this setting, based on a recent technique for recording quantum queries. As an application, we consider the task of finding multiple collision pairs in a random function. We show that the quantum query complexity of this problem increases when the available space decreases.

Keywords: quantum computing, algorithms, Monte Carlo method, mean estimators, importance sampling, stochastic optimization, space-bounded computation, streaming algorithms, query complexity.

Résumé

La méthode de Monte Carlo est un paradigme central de l'algorithmique basée sur l'analyse statistique et sur les techniques d'échantillonnage aléatoire appliquées aux problèmes ayant une interprétation probabiliste. Cette thèse explore les avantages qu'offrirait un ordinateur quantique pour augmenter l'efficacité de cette méthode.

Nous étudions dans un premier temps le problème d'estimation de valeurs moyennes par des techniques à temps de calcul minimal. Nous développons de nouveaux algorithmes quantiques pour estimer l'espérance d'une variable aléatoire réelle produite en sortie d'un processus quantique. Les estimateurs que nous construisons procurent une accélération quadratique par rapport au nombre d'échantillons classiques nécessaires pour obtenir une borne d'erreur sous-gaussienne ou une (ϵ, δ) -approximation. Nous décrivons également un cadre théorique fournissant une notion de «temps d'arrêt» pour un processus quantique générant une variable aléatoire. Nous démontrons que le problème d'estimation de la moyenne peut être résolu plus efficacement lorsque le temps d'arrêt moyen du processus sous-jacent est court. Ces résultats sont appliqués au développement d'un algorithme de requête quantique quasi-optimal pour approximer le nombre de triangles dans un graphe.

Dans un second temps, nous étudions le problème d'échantillonnage préférentiel et ses applications en optimisation stochastique. Nous construisons un algorithme quantique pour échantillonner plusieurs éléments d'une distribution finie spécifiée par un vecteur de probabilité. Notre approche résout le problème plus général consistant à préparer plusieurs copies d'un état quantique dont les amplitudes sont accessibles via un oracle. L'utilité de ce résultat est illustrée à travers le développement de deux algorithmes hybrides quantiques-classiques basés sur la méthode des poids multiplicatifs et sur l'algorithme du gradient stochastique. Ces deux applications concernent la prédiction en ligne avec conseil d'experts, et la minimisation des fonctions sous-modulaires.

La dernière partie de cette thèse est consacrée à l'étude des algorithmes quantiques à mémoire limitée. Nous étudions tout d'abord le problème d'approximation des moments de fréquence dans le modèle de flots de données à passes multiples. Nous construisons un algorithme quantique qui nécessite une quantité de mémoire moindre que les meilleurs algorithmes de streaming classiques possible. Notre méthode repose sur les estimateurs quantiques de moyenne susmentionnés et sur des techniques de simulation de calcul réversible. Nous explorons ensuite certaines limites des algorithmes à mémoire restreinte dans le modèle de requête quantique. Nous développons une nouvelle approche pour obtenir des bornes inférieures temps-mémoire, basée sur une technique récente d'enregistrement des requêtes quantiques. Ce résultat est appliqué au problème de la recherche de paires de collisions dans une fonction aléatoire. Nous démontrons que la complexité en requête quantique de cette tâche augmente lorsque la quantité de mémoire disponible diminue.

Mots clés : calcul quantique, algorithmes, méthode de Monte Carlo, estimateurs de moyenne, échantillonnage préférentiel, optimisation stochastique, calcul à mémoire restreinte, algorithmes de streaming, complexité en requête.

Acknowledgments

First and foremost, I want to express my deepest gratitude to my supervisors, Frédéric Magniez and Miklos Santha, for their kindness, trust and support throughout my PhD. I had the privilege to work in two extraordinary places, the *Institut de Recherche en Informatique Fondamentale* in Paris and the *Centre for Quantum Technologies* in Singapore, where they always made me feel welcome and valued. I want to thank them especially for the time they spent sharing their experience and knowledge with me, which helped me grow as a researcher and as a person.

I am thankful to Anil Ada, Olivier Bournez, Hartmut Klauck, Sophie Laplante and Roberto Mantaci for hosting me as an intern during my bachelor's and master's studies. They guided my first steps into research and made me feel confident enough to engage in a PhD. I owe a special thanks to Omar Fawzi, who introduced me to quantum computing as a teacher, and helped me find my path in the quantum community.

I am grateful to Stacey Jeffery, Patrick Rebentrost, Ansis Rosmanis, Thomas Vidick and Ronald de Wolf for the enlightening discussions and collaborations we had together. I would like to thank Ashley Montanaro and Ashwin Nayak for inviting me to present my work at the Institute for Quantum Computing and the University of Bristol. I also want to express my gratitude to Omar Fawzi, Stacey Jeffery, Ashley Montanaro, Michele Mosca and María Naya-Plasencia for being on my thesis committee.

My PhD experience wouldn't have been complete without colleagues to play board games, drink coffee and travel with. For that, I want to thank Alessandro, Alex, Alexandre, Amaury, Anupa, Arjan, Daniel, Etienne, Jonas, Maharshi, Mickaël, Olivier, Sander, Sidi Mohamed, Simon, Simona, Xavier, Yixin, Zhouningxin, and my other fellow students. I hope our paths cross again!

Finally, I want to thank my family and friends for their invaluable contributions to this work. I'd like to give special thanks to Louisa, Amine, Nabil and Zakia for their hospitality during my stay in Paris, Sarah for sharing her daily stories and ever-growing cooking skills with me, Marwan and Driss for our too-rare hiking and biking trips, and Bassem and Thomas for their long-lasting friendship. I dedicate this thesis to my parents, may it be a blessing to them.

Contents

1	Introduction	1
1.1	Preliminaries: algorithmic model	2
1.2	Quantum algorithms for estimating average values	3
1.3	Quantum algorithms for optimization with importance sampling	4
1.4	Quantum algorithms with limited memory	6
I	Preliminaries	9
2	Mathematical Preliminaries	11
2.1	Linear algebra and notations	11
2.2	Concentration inequalities	12
3	Algorithmic Preliminaries	15
3.1	Quantum circuit model	15
3.2	Amplitude amplification	18
3.3	Amplitude estimation	20
II	Quantum Algorithms for Estimating Average Values	23
4	Mean Estimation Problem	25
4.1	Introduction	25
4.1.1	Related work	26
4.1.2	Contributions and organization	27
4.1.3	Proof overview	28
4.2	Model of input	30
4.3	Quantile estimation	31
4.4	Sub-Gaussian estimator	34
4.5	(ϵ, δ) -Estimators	36
4.5.1	Parameter-free estimators	37
4.5.2	Parametrization by the coefficient of variation	38
4.6	Lower bounds	41
4.6.1	Sub-Gaussian estimation	41
4.6.2	(ϵ, δ) -Estimation	42
4.6.3	State-based estimation	42
4.7	Discussion	43
5	Variable-Time Mean Estimation	45
5.1	Introduction	45
5.1.1	Related work	46
5.1.2	Contributions and organization	46
5.1.3	Proof overview	47

5.2	Model of input	48
5.3	Variable-time amplitude estimation	50
5.3.1	Notations	50
5.3.2	State generation algorithms	51
5.3.3	Main algorithm	53
5.4	Variable-time mean estimator	55
5.4.1	Variable-time Bernoulli estimator	55
5.4.2	Variable-time (ϵ, δ) -estimator	57
5.5	Discussion	60
6	Estimation of Graph Parameters	61
6.1	Introduction	61
6.1.1	Related work	62
6.1.2	Contributions and organization	62
6.1.3	Proof overview	63
6.2	Preliminaries	67
6.3	Edge counting	68
6.4	Triangle counting	69
6.4.1	Assumptions	69
6.4.2	Main concepts	71
6.4.3	Triangle degree estimator	73
6.4.4	Weighted triangle degree estimator	75
6.4.5	Final algorithm	80
6.5	Lower bounds	81
6.6	Discussion	83
III	Quantum Algorithms for Optimization with Importance Sampling	85
7	Quantum State Preparation and Importance Sampling	87
7.1	Introduction	87
7.1.1	Related work	88
7.1.2	Contributions and organization	89
7.1.3	Proof overview	89
7.2	Model of input	90
7.3	Preliminaries	90
7.4	Preparing K copies of a quantum state	91
7.5	Preparing K samples from a discrete distribution	93
7.6	Discussion	95
8	Applications to Stochastic Optimization	97
8.1	Introduction	97
8.1.1	Related work	98
8.2	Hedge algorithm	99
8.2.1	Classical Hedge algorithm	99
8.2.2	Quantum Hedge algorithm	100
8.3	Submodular function minimization	102
8.3.1	Proof overview	103
8.3.2	Preliminaries	103
8.3.3	Data structures and c -covers	105

8.3.4	Importance sampling for gradient computation	107
8.3.5	Final algorithm	109
8.4	Discussion	112
IV	Quantum Algorithms with Limited Memory	113
9	Frequency Moments and Linear Sketches in the Data Stream Model	115
9.1	Introduction	115
9.1.1	Related work	116
9.1.2	Contributions and organization	117
9.1.3	Proof overview	117
9.2	Data stream model	118
9.3	Quantum simulation of classical streaming algorithms	119
9.3.1	Reversible streaming algorithms	119
9.3.2	Linear sketch algorithms	121
9.4	Estimation of the frequency moments	122
9.5	Discussion	124
10	Time-Space Tradeoffs by Recording Queries	125
10.1	Introduction	125
10.1.1	Related work	126
10.1.2	Contributions and organization	127
10.1.3	Proof overview	128
10.2	Models of computation	130
10.2.1	Query model	130
10.2.2	Space-bounded model	131
10.3	Recording model	131
10.4	Time lower bound for Collision Pairs Finding	133
10.4.1	Recording query operator	133
10.4.2	Analysis of the recording progress	134
10.4.3	From the recording progress to the success probability	135
10.5	Time lower bound for K -Search	137
10.5.1	Recording query operator	137
10.5.2	Analysis of the recording progress	138
10.5.3	From the recording progress to the success probability	139
10.6	Time-space tradeoffs	140
10.6.1	Time-space tradeoff for Collision Pairs Finding	140
10.6.2	Time-space tradeoff for Sorting	144
10.7	Discussion	144
	Bibliography	145

1

Introduction

The development of the Monte Carlo method went hand in hand with the construction of the first electronic digital computers in the late 1940s. Ulam and von Neumann rapidly saw the potential applications of these machines to the field of nuclear physics. This new computational power, coupled with statistical sampling techniques, was soon used to study the motions and interactions of neutrons. In a pioneering letter (see [And86; Met87; Eck87] for a brief history of the Monte Carlo method), von Neumann outlined a computer program that simulates nuclear chain reactions based on computer-generated random numbers and statistical analysis. Nowadays, the Monte Carlo method refers to a broad class of algorithms that use randomness, statistics and approximation methods to solve computational problems. Its scope of application has expanded from statistical physics to mathematical finance, operational research, Bayesian statistics, and other areas (see [KBTB14] for an account of its importance in modern sciences, finance and engineering). The Monte Carlo method revolves around two fundamental ideas, which occupy a central place in this thesis. The first one is to use approximation techniques to estimate a numerical parameter whose exact computation is difficult. The objective here is to trade off accuracy against efficiency in an optimal way. The second idea is to let an algorithm be guided by random choices with the goal that it outperforms deterministic strategies. This requires the design of random processes that can sample from specific probability distributions. A toy example that illustrates the Monte Carlo method is the following random experiment for estimating the Euler number e . Sample a sequence of uniform random numbers x_1, x_2, \dots in the interval $[0, 1]$ until the first time T when the sum $\sum_{i=1}^T x_i$ is larger than 1. One can show [Rus91] that the value of T is an unbiased estimate of e .

About 30 years after the emergence of the Monte Carlo method, in a study of the Closest Pair of Points problem, Rabin [Rab76] formulated what is often considered one of the first randomized algorithms. Randomness became a useful resource not only to simulate physical processes but also to solve purely algorithmic problems. It shaped our modern conception of algorithm designs, and it is now a central topic in computer science with a plethora of applications [MR95; MU17]. Another revolution occurred in the 1980s, driven by the ambition of simulating quantum physics problems. Feynman [Fey82; Fey86] observed that such simulations tend to require a prohibitive amount of resources when run on a conventional computer. Together with other physicists, he envisioned a new model of computation, the *quantum computer*, that would exploit the laws of quantum mechanics to surpass these barriers. This hypothetical machine operates on a new unit of information, the quantum bit (or *qubit*), and it leverages the principles of quantum mechanics (superposition, entanglement, etc.) to perform the computation. The theoretical basis of this model was laid by Deutsch [Deu85; Deu89], who formalized the notions of quantum Turing machines and quantum circuits. The early quantum algorithms discovered

by Deutsch and Jozsa [DJ92], Simon [Sim97], and Bernstein and Vazirani [BV97] opened a new path for quantum speedups over classical algorithms. The groundbreaking algorithm of Shor [Sho97] for integer factorization was the spark that triggered the rapid growth of this field of research. It was followed by another important algorithm from Grover [Gro96a] for searching an item in an unordered list. Nowadays, quantum algorithm design has evolved into a mature topic with applications in linear algebra, Hamiltonian simulation, optimization, machine learning, etc. In particular, the techniques and motivations behind the Monte Carlo method are now revisited through the lens of quantum mechanics with the goal of finding new applications and more efficient algorithms. In this thesis, we contribute to this line of research by studying the power and limitations of this approach.

1.1 Preliminaries: algorithmic model

Before diving into the description of our contributions, we describe the general features of the quantum algorithms studied in this thesis. We present the main algorithmic paradigm employed in our results, and we highlight the general input-output model.

Quantum algorithmic paradigm. There are a few numbers of quantum subroutines (Quantum Fourier Transform, Phase Estimation, Grover’s Search, etc.) that are part of most quantum algorithms. In this thesis, our approach is rooted in the use of the *Grover operator* described in the seminal work of Grover on quantum search [Gro96a]. The Grover operator is a unitary transformation that rotates the state vector of a quantum system in a certain two-dimensional space, and that can be efficiently simulated on a quantum computer in specific scenarios. In the present work, the Grover operator is embedded into *hybrid quantum-classical algorithms*. This approach follows an active line of research that consists of combining advanced classical algorithms with quantum subroutines to go beyond the “off-the-shelf” applications of the latter. These methods offer a speedup that is often polynomial over the best possible randomized algorithms, as is the case in this thesis.

Input. We work in the *black-box model*, where the input to a problem is provided by an oracle whose inner workings are not accessible to the algorithm. An oracle is represented as a unitary operator that returns a piece of information about the input whenever it is applied to a state vector. A canonical example is to encode an input Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ as a bijective function oracle mapping $(x, y) \mapsto (x, y \oplus f(x))$. The way this transformation is implemented is left unspecified (it may be given to us as a Boolean circuit, for instance). The black-box model is a convenient tool when one wants to abstract the role of a subroutine (the oracle) and focus only on the number of times it is invoked (or queried). Most of the known quantum algorithms can be seen as acting in this setting. For instance, Grover’s search uses $O(2^{n/2})$ queries to a function oracle to find a value x such that $f(x) = 1$. In this thesis, we consider both function oracles and more general oracles selected from a predefined set of unitaries. We make an exception in Chapter 9, where we use a non-black-box input model (the data stream model).

Output. Our quantum algorithms return classical results, except in Chapter 7 where we produce a quantum state. We often look for approximate solutions that do not deviate significantly from the best possible output. We also allow the algorithms to be incorrect with a small probability.

1.2 Quantum algorithms for estimating average values

The problem of finding efficient algorithms to estimate statistics lies at the core of the Monte Carlo method. A standard approach is to express a parameter of interest as the expectation of a random variable X (ideally with low variance) and to gather sufficient samples from X to construct an estimate of $\mathbb{E}[X]$. The challenge here is twofold. First, the experiment that generates the samples must be efficiently computable. Secondly, one needs a mean estimator to process the samples and return a high-accuracy estimate of $\mathbb{E}[X]$. This approach is often known as *approximate counting* in computer science. It is a method of choice to estimate parameters whose exact computation is $\#P$ -hard, such as the number of satisfying assignments to a DNF [KL83; KLM89], the volume of convex bodies [DFK91; DF91] or the permanent [JSV04]. More generally, it is used when exact counting is intractable—for example, in high-dimensional integration, in Bayesian inference, or for estimating partition functions in statistical physics. In parallel to these works, approximate counting has been studied in the scenario where the amount of available computational resources (time, space, etc.) is *sublinear* in the input size of the problem. In a seminal work, Feige [Fei06] showed, for instance, that the average degree in a graph can be estimated with a number of degree queries that is sublinear in the number of vertices. This result prompted the study of sublinear-time algorithms for subgraph counting problems. In another pioneering use of this method, Alon, Matias and Szegedy [AMS99] designed an algorithm to estimate statistics of a stream of data by using an amount of memory that is sublinear in the stream length. This result popularized the field of streaming algorithms (we return to the data stream model in Section 1.4).

The first quantum algorithms for estimating statistics were found in the wake of Grover’s search. Several quantum speedups were obtained to compute the minimum [DH96], the median [Gro96b; Gro98; NW99] or the k -th smallest element [NW99; DHHM06] of a list of N numbers. The task of *quantum counting* was introduced by Brassard, Høyer and Tapp in [BHT98a], with the goal of (approximately) counting the number of ones in a list of N Boolean values. This result has evolved into more advanced algorithms for numerical integration [AW99; Nov01; Hei02; TW02] and estimating partition functions [WCNA09; PW09; Mon15; HW20]. Nevertheless, the power of quantum computing for solving approximate counting problems is much less understood than that of classical computing. This is due in part to the fact that standard mean estimators (such as the empirical mean or the median-of-means) and concentration inequalities (such as Chebyshev’s inequality) have no clear equivalent when given “quantum access” to a random variable. There do exist quantum estimators for estimating the mean of N real numbers [Gro98; Ter99; BDGT11; Mon15]. However, they are outperformed by classical estimators if the distribution is heavy tailed, or they require additional information on the input (e.g. the variance).

Our contributions. Part II of this thesis is devoted to estimating average values in a sample and time-efficient way.

In Chapter 4, we consider the general problem of estimating the mean μ of a real-valued random variable X . We suppose that X is obtained as the output of a separate quantum computation, and we introduce the concept of *quantum experiment* to model the task of getting one sample of information from X . We seek to minimize the number of quantum experiments needed to estimate μ with some prescribed accuracy, under the assumption that X has a bounded variance σ^2 . In the classical setting, the optimal deviation bound is achieved by the so-called *sub-Gaussian estimators* that return an estimate $\tilde{\mu}$ such that $|\tilde{\mu} - \mu| \leq O(\sigma\sqrt{\log(1/\delta)/t})$ with probability at least $1 - \delta$, where t denotes the number of

i.i.d. samples available to the estimator. We present a quantum estimator that achieves the bound $|\tilde{\mu} - \mu| \leq \tilde{O}(\sigma \log(1/\delta)/t)$ while performing only t quantum experiments. This represents a near-quadratic and optimal speedup over the classical sub-Gaussian estimators. As an application, we obtain new quantum algorithms for the (ϵ, δ) -approximation problem where the goal is to satisfy $|\tilde{\mu} - \mu| \leq \epsilon|\mu|$ with probability at least $1 - \delta$.

In [Chapter 5](#), we refine the definition of a quantum experiment to introduce the notion of *stopping time*. The stopping time T is a random variable distributed according to the “variable time” spent by an experiment to perform its computation. The variable time of a quantum algorithm is a concept introduced by Ambainis [[Amb12](#)] in the design of the *variable-time amplitude amplification* algorithm. The existing quantum mean estimators use on the order of $N \times \max(T)$ operations to perform N quantum experiments. This contrasts with the classical setting, where N samples are obtained in *expected* time $N \times \mathbb{E}[T]$. This difference is explained by the fact that a quantum experiment prepares a coherent encoding of X , which could be destroyed if a measurement is applied before the completion time $\max(T)$. We manage to overcome this obstacle by giving a quantum estimator for the (ϵ, δ) -approximation problem that uses on the order of $\mathcal{V} \times \sqrt{\mathbb{E}[T^2]}$ operations, where \mathcal{V} is quadratically smaller than the number of samples needed classically. Our technique is rooted in the development of a new *variable-time amplitude estimation* algorithm.

In [Chapter 6](#), we use the techniques developed in the two previous chapters to study the problem of approximating the number of *edges* and *triangles* in a graph in sublinear time. We consider the general graph model where neighbor and vertex-pair quantum queries are permitted. Given a graph with n vertices, m edges and t triangles, our algorithms return an edge estimate \tilde{m} such $|\tilde{m} - m| \leq \epsilon m$ after $\tilde{O}(\sqrt{n}/m^{1/4})$ queries (omitting the dependence on the relative error ϵ), and a triangle estimate \tilde{t} such that $|\tilde{t} - t| \leq \epsilon t$ after $\tilde{O}(\sqrt{n}/t^{1/6} + m^{3/4}/\sqrt{t})$ queries. This is better than the best possible classical algorithms. Our algorithms consist of estimating the expectations of carefully chosen random variables. The latter are generated by using local graph exploration techniques adapted from the recent classical triangle counting algorithm of Eden, Levi, Ron and Seshadhri [[ELRS17](#)].

1.3 Quantum algorithms for optimization with importance sampling

The Monte Carlo method offers a large variety of techniques for sampling from a sophisticated distribution [[RC04](#)]. Two paramount examples are *rejection sampling* and *importance sampling*, which resort to proxy distributions. The rejection sampling method (also called acceptance-rejection sampling) can be found in the work of von Neumann on random number generations [[Neu51](#)]. It simulates a target distribution $p(i)$ by sampling from another distribution $q(i)$ and keeping the result with probability $\frac{p(i)}{mq(i)}$ (where m is a constant that maintains the ratio below 1). The importance sampling method can be traced back to the papers of Kahn [[Kah50a](#); [Kah50b](#)] on radiation shielding. It consists of modifying a distribution such that the most important events become more likely to occur. The canonical example (used in numerical integration, for instance) is to reweight the expectation $\mathbb{E}_p[f(i)]$ of a function f under a distribution p into the expectation $\mathbb{E}_q[f(i) \frac{p(i)}{q(i)}]$ under a new *importance* distribution q . The choice of q is made so that it is easy to sample from it, while lowering the variance. Ideally, $q(i)$ ought to be proportional to the weight $|f(i)|p(i)$. A major alternative to rejection and importance sampling, which is well suited for high-dimensional distributions, is the class of *Markov Chain Monte Carlo* methods, which have started with the Metropolis—Hastings algorithm [[MRR+53](#); [Has70](#)].

The problem of sampling from a distribution is generalized in the quantum model to the problem of preparing a *q-sample*. A q-sample is a quantum state that encodes the target distribution as a coherent superposition over the sample space. There are a few techniques based on the Monte Carlo method for preparing a q-sample (see [HW20] and references therein). One of the first algorithms was given by Grover [Gro00b], who adapted the rejection sampling method to transform the amplitudes of a uniform superposition into that of a target state $\sum_i \sqrt{p(i)}|i\rangle$. Ozols, Roetteler and Roland [ORR13] generalized this method to start with any prior state $\sum_i \sqrt{q(i)}|i\rangle$. The theory of quantum walks [Sze04] has also led to quantum speedups for certain Markov Chain Monte Carlo methods, such as simulated annealing [SBBK08; WA08; OBD18; HW20]. Understanding the power and limitations of q-samples is still a largely open question. On one hand, Bshouty and Jackson [BJ99] and Aharonov and Ta-Shma [AT07] obtained quantum speedups over the best existing classical algorithms for learning a DNF under the uniform distribution or solving the problems in the complexity class SZK (such as graph isomorphism). On the other hand, Arunachalam and de Wolf [AW18] proved that learning a function in the general PAC model requires as many q-samples as classical samples. This latter result encourages one to search for quantum speedups at a different stage of the computation (e.g. when preparing the q-samples that are fed to a quantum learner).

Another use of q-samples is to *classically* sample from a distribution by preparing and measuring the relevant quantum state. This approach is particularly well suited for *stochastic optimization*, where sampling-based Monte Carlo methods are ubiquitous. There, the goal is to minimize an objective function $\min_{\theta} f(\theta)$ guided by a random process. Homem-de-Mello and Bayraksan [HB14] provided an extensive review of the use of Monte Carlo sampling-based methods in this setting. Among other advantages, it can reduce the computational time or help to escape a local optimum. These methods also apply to deterministic problems for which there is a probabilistic interpretation. A major example is to rephrase a sum of loss functions $f(\theta) = \sum_{i=1}^N f_i(\theta)$ as a stochastic objective function $f(\theta) = \mathbb{E}_{\mathcal{U}}[N f_i(\theta)]$ under the uniform distribution and to estimate it by importance sampling. This idea is nicely illustrated by the randomized Kaczmarz algorithm of Strohmer and Vershynin [SV09] for solving linear systems. It raises the question of how to balance the effort between sampling from the chosen importance distribution and iterating toward a minimum.

Our contributions. Part III of this thesis is devoted to the use of quantum state preparation for importance sampling and stochastic optimization.

In Chapter 7, we study the problem of obtaining *multiple* i.i.d. samples from an importance distribution. We consider the case where the input is specified by an *arbitrary* weight vector (w_1, \dots, w_N) that can be queried in superposition. The goal is to sample from the distribution that returns $i \in [N]$ with probability $|w_i|/W$, where W is the (unknown) normalization factor. This setting represents the case where no prior information is known about the chosen importance distribution. Grover [Gro00b] constructed an algorithm that samples a single element from that distribution in time $O(\sqrt{N})$. In fact, he solved the more general problem of preparing the q-sample $|w\rangle := \sum_i \sqrt{|w_i|/W}|i\rangle$. In practice, it is reasonable to assume that several samples or copies of $|w\rangle$ are needed for further use (such a case occurs in Chapter 8). Thus, a natural question is whether the average preparation cost per state can be made smaller than $O(\sqrt{N})$. We answer this question positively by constructing an optimal algorithm that prepares the K -fold state $|w\rangle^{\otimes K}$ in time $O(\sqrt{KN})$ for any $K \geq 1$. Our technique uses a refinement of the quantum rejection sampling method employed by Grover.

In [Chapter 8](#), we use the quantum importance sampling algorithm to develop two independent hybrid quantum-classical algorithms for stochastic optimization. Our first algorithm addresses the problem of *online prediction with expert advice*. Consider a game with T rounds where we play a mixture of N strategies in each round and observe the loss incurred by this choice in the next round. The Hedge algorithm by Freund and Schapire [[FS97](#)] guarantees an optimal regret bound by using a multiplicative weight update method in total time $O(TN)$. We present a quantum algorithm achieving a quantum speedup in N while the overall regret remains close to that of the Hedge algorithm with high probability. Our algorithm is a simple modification of the Hedge algorithm that consists of choosing the strategy by quantum importance sampling. Our second application is more involved and concerns the problem of minimizing a *submodular* set function. Submodular functions are set functions mapping every subset of some ground set of size n into the real numbers and satisfying the diminishing returns property. Submodular minimization is an important field in discrete optimization theory due to its relevance to various branches of mathematics, computer science and economics. In a recent paper, Chakrabarty et al. [[CLSW17](#)] constructed the first subquadratic algorithm for finding an approximate minimum with additive error ϵ in time $\tilde{O}(n^{5/3}/\epsilon^2)$. We present a quantum algorithm that improves upon this result by running in time $\tilde{O}(n^{5/4}/\epsilon^{5/2})$. Our technique consists of using a stochastic subgradient descent method, where the subgradient directions are obtained by quantum importance sampling.

1.4 Quantum algorithms with limited memory

The Monte Carlo method is often used to process massive datasets, such as those generated by experiments in particle physics (e.g. the Large Hadron Collider) or by web traffic. In this case, the amount of workspace available is much smaller than the size of the inputs, which motivates the search for algorithms that can operate with *limited memory*. The study of space-bounded computations has been carried out in a variety of models. For instance, in a pioneering work, Munro and Paterson [[MP78](#)] studied the amount of working memory needed by an algorithm that makes a limited number of passes over a one-way read-only input tape. This was later formalized as the *data stream model*, where the input arrives as a long stream of data that cannot fit entirely in memory. Here, the objective is to compute some relevant statistics about the stream, such as the most frequent element or the median value. The use of randomness and approximation methods is an essential ingredient to decrease space usage. As an example, the randomized *approximate counting algorithm* of Morris and Flajolet [[Mor78](#); [Fla85](#)] can estimate the total number n of elements in a stream by using only $O(\log \log n)$ bits of memory. In cryptography, the statistical properties of random mappings [[FO89](#)] play a central role in reduced memory attacks, such as Pollard's rho algorithm for integer factorization [[Pol75](#)] or Hellman's time-memory tradeoff for function inversion [[Hel80](#)]. In many cases, decreasing the memory size comes at the cost of increasing the use of other computational resources (e.g. the running time, the number of passes over a stream, etc.). Understanding the inherent *tradeoffs* between space and other complexity measures is a far-reaching problem in complexity theory. To name one recent example, Raz [[Raz18](#)] proved that for some learning problems, a small memory must imply a long learning process.

The storage of information in quantum systems obeys fundamentally different principles than the storage of information in classical ones (no-cloning, superposition, uncertainty, etc.). Moreover, memory will be a critical resource in near-term quantum computers. Thus, it is a major task to understand the properties of space-bounded quantum computations.

There are some examples where quantum mechanics help to use less memory. In the data stream model, Le Gall [Gal09] described, for instance, an artificial problem for which a quantum computer would provide *exponential* savings in memory over the best possible classical algorithm. Ta-Shma [TS13] identified several tasks (such as matrix inversion) in the quantum Turing machine model that can be solved in quantum logspace, whereas the best known classical algorithms use quadratically more memory. Nevertheless, quantum computation is a mixed blessing with regard to memory. For instance, *reversibility* is a core property used in quantum algorithm design, but the standard techniques to make a computation reversible (such as the deferred measurement principle [AKN98] or Bennett’s reversible simulation [Ben73]) often require expanding the workspace. For certain problems, the fastest known quantum algorithms [BHT98b; Amb07; LZ19a] use much more memory than the classical ones. A central open question is whether a speedup both in terms of time and space complexities is achievable for such problems.

Our contributions. Part IV of this thesis is devoted to the study of space-bounded quantum computation in two different models of computation: the data stream model and the quantum query model.

In Chapter 9, we consider the problem of estimating the *frequency moments* F_k of a stream. The frequency moments are important statistics introduced by Alon, Matias and Szegedy [AMS99] in a pioneering work on the data stream model. The space complexity of approximating F_k is the object of rich literature that led to the development of major streaming algorithms. It culminated in the proof that $S = \tilde{\Theta}(n^{1-2/k}/P)$ is the optimal memory size needed for estimating F_k on a length- n input with a classical algorithm making P passes over a stream [MW10; AKO10; WZ12]. We demonstrate that the use of a quantum computer can decrease the memory size to $S = \tilde{O}(n^{1-2/k}/P^2)$ while keeping the number of passes to P . There are very few quantum speedups known in the data stream model due to the sequential and uncontrolled access to the input. Our results contribute to giving new quantum algorithmic methods in this model. In particular, we describe a general *reversible* simulation technique that applies to a broad class of classical streaming algorithms known as *linear sketches*. We combine this technique with the quantum mean estimator developed in the early parts of this thesis to construct our algorithm.

In Chapter 10, we study *time-space tradeoff* results to investigate how much time T is needed to solve a particular task when only S qubits of memory are available. We work in the quantum circuit model with query access to the input. Our main contribution is a new and simple approach for proving *lower bounds* in this setting, based on the recent recording query technique of Zhandry [Zha19]. The very few existing quantum time-space tradeoff lower bounds [KŠW07; AŠW09] require heavy use of the adversary or polynomial methods. As the first application of our technique, we consider the problem of finding multiple collision pairs in a random hash function. This question plays a central role in cryptography, notably for meet-in-the-middle attacks. There is a long-standing conjecture on the need for a large quantum memory to find one collision pair faster than with the classical Pollard rho method. We prove that, for the related problem of finding K collision pairs in a random function $f : [N] \rightarrow [N]$, one has to perform at least $T \geq \Omega(K(N/S)^{1/3})$ quantum queries. On the other hand, we show that the optimal algorithm with unlimited memory uses $T = \tilde{\Theta}(K^{2/3}N^{1/3})$ queries. These results give the first evidence that limiting the size S of the available quantum memory makes the problem of finding collisions harder to solve. As a second application, we give a simpler proof of the time-space tradeoff $T^2S \geq \Omega(N^3)$ for sorting N numbers on a quantum computer, which was first obtained by Klauck, Špalek and de Wolf [KŠW07].

Part I

Preliminaries

2

Mathematical Preliminaries

2.1 Linear algebra and notations

Given the n -dimensional Hilbert space $\mathcal{H} = \mathbb{C}^n$, we let $|\psi\rangle \in \mathcal{H}$ denote a vector in \mathcal{H} , and we let $\langle\psi|$ denote the conjugate transpose of $|\psi\rangle$. The inner product between two vectors $|\psi\rangle, |\phi\rangle \in \mathcal{H}$ is represented as $\langle\psi|\phi\rangle \in \mathbb{C}$ and the outer product is $|\psi\rangle\langle\phi| \in \mathbb{C}^{n \times n}$. The standard basis of \mathcal{H} (also called the *computational basis*) is denoted by $(|0\rangle, |1\rangle, \dots, |n-1\rangle)$. Thus, any vector $|\psi\rangle \in \mathcal{H}$ can be written as

$$|\psi\rangle = \sum_{i=0}^{n-1} \alpha_i |i\rangle$$

where $\alpha_i = \langle i|\psi\rangle$ for $i = 0, \dots, n-1$. The norm $\sqrt{\sum_{i=0}^{n-1} |\alpha_i|^2}$ of $|\psi\rangle$ is denoted by $\| |\psi\rangle \|$ or $\|\psi\|$. The induced matrix norm is the *spectral norm*, written as $\|U\|$ for a linear operator $U : \mathcal{H} \rightarrow \mathcal{H}$. We let $\mathcal{H}_1 \otimes \mathcal{H}_2$, $|\psi_1\rangle \otimes |\psi_2\rangle$ and $U_1 \otimes U_2$ denote the tensor products of, respectively, two Hilbert spaces $\mathcal{H}_1 = \mathbb{C}^n, \mathcal{H}_2 = \mathbb{C}^m$, two vectors $|\psi_1\rangle \in \mathcal{H}_1, |\psi_2\rangle \in \mathcal{H}_2$, and two linear operators $U_1 : \mathcal{H}_1 \rightarrow \mathcal{H}_1, U_2 : \mathcal{H}_2 \rightarrow \mathcal{H}_2$. Note that $(U_1 \otimes U_2)(|\psi_1\rangle \otimes |\psi_2\rangle) = (U_1|\psi_1\rangle) \otimes (U_2|\psi_2\rangle) \in \mathcal{H}_1 \otimes \mathcal{H}_2$. The vector $|\psi_1\rangle \otimes |\psi_2\rangle$ is also denoted by $|\psi_1\rangle|\psi_2\rangle$. The standard basis of $\mathcal{H}_1 \otimes \mathcal{H}_2$ is identified with $(|i\rangle|j\rangle)_{0 \leq i < n, 0 \leq j < m}$, and we equivalently write $|i, j\rangle$ or $|ij\rangle$ for the basis states (the latter notation is often used when i and j are expressed in base two).

Qubits. We use the standard formalism for describing *pure* quantum states. A *one-qubit state* is represented as a vector $|\psi\rangle \in \mathbb{C}^2$ with unit norm $\|\psi\| = 1$. A system made of n qubits is called an *n -bit quantum register* and it is described by an *n -qubit state*, that is a vector $|\psi\rangle \in \mathbb{C}^{2^n}$ with unit norm $\|\psi\| = 1$. The qubit state $|0\rangle^{\otimes n}$ is also denoted by $|0^n\rangle$ or $|\mathbf{0}\rangle$ when n is clear from the context. Given two qubit states $|\psi\rangle, |\phi\rangle \in \mathbb{C}^{2^n}$, the value of $\langle\phi|\psi\rangle$ is the *amplitude* of the state $|\phi\rangle$ in $|\psi\rangle$. Given a qubit state $|\psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ where $\mathcal{H}_1, \mathcal{H}_2$ are two Hilbert spaces, we say that $|\psi\rangle$ is a *product state* if there exist $|\psi_1\rangle \in \mathcal{H}_1, |\psi_2\rangle \in \mathcal{H}_2$ such that $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, and otherwise we say that $|\psi\rangle$ is an *entangled state*.

Operators. The *identity* operator over the Hilbert space $\mathcal{H} = \mathbb{C}^n$ is denoted by I or I_n . A *unitary* operator U is a linear map $U : \mathcal{H} \rightarrow \mathcal{H}$ that satisfies $U^\dagger U = U U^\dagger = I$, where U^\dagger is the conjugate transpose of U . Note that the inverse of a unitary operator is $U^{-1} = U^\dagger$. An (orthogonal) *projector* $\Pi : \mathcal{H} \rightarrow \mathcal{H}$ is a linear operator that satisfies $\Pi^2 = \Pi = \Pi^\dagger$. A particular example is the projector $|\psi\rangle\langle\psi|$, for $|\psi\rangle \in \mathcal{H}$, that projects on the one-dimensional subspace of \mathcal{H} spanned by $|\psi\rangle$. Two projectors Π_1, Π_2 acting on \mathcal{H} are

orthogonal if $\Pi_1 \Pi_2 = 0$. Finally, a unitary operator $U : \mathcal{H} \rightarrow \mathcal{H}$ is a *reflection* if $U^2 = I$. One can check that $I - 2\Pi$ is a reflection when Π is a projector.

General notations. Given an integer $n \in \mathbb{N}$, we define $[n] = \{1, \dots, n\}$. The set of all non-negative (resp. non-positive) real numbers is denoted by $\mathbb{R}_{\geq 0}$ (resp. $\mathbb{R}_{\leq 0}$). Given $x \in \mathbb{R}$, we define $\text{sgn}(x)$ to be 1 if $x \geq 0$, and -1 otherwise. We let $\mathbb{1}_P \in \{0, 1\}$ denote the Boolean value that equals 1 if and only if the predicate P is true. Given two sets R and D we let R^D denote the set of all functions $F : D \rightarrow R$. We use the asymptotic notation $f(n) = \tilde{O}(g(n))$ to indicate that $f(n) = O(g(n) \log^k g(n))$ for some (not necessarily positive) constant k independent of n . We similarly define $f(n) = \tilde{\Omega}(g(n))$ when $f(n) = \Omega(g(n) \log^k g(n))$, and $f(n) = \tilde{\Theta}(g(n))$ when $f(n) = \Theta(g(n) \log^k g(n))$.

Notations for real-valued vectors. We let $e_i \in \mathbb{R}^n$ be the standard basis state with a 1 at position $i \in [n]$ and 0 elsewhere. Given a vector $u = (u_1, \dots, u_n) \in \mathbb{R}^n$ and an integer $p \geq 0$, we let $\|u\|_p = (\sum_{i \in [n]} |u_i|^p)^{1/p}$ denote the ℓ_p -norm of u . The largest entry in u (in absolute value) is denoted by $\|u\|_\infty = \max_{i \in [n]} |u_i|$. We say that u is k -sparse if it contains at most k non-zero entries. We define the non-negative $u_{\geq 0} \in \mathbb{R}_{\geq 0}^n$ and the non-positive $u_{\leq 0} \in \mathbb{R}_{\leq 0}^n$ parts of u as the vectors with disjoint supports satisfying $u = u_{\geq 0} + u_{\leq 0}$. Given a set $S \subseteq [n]$, we define the vector $u_S = (u_i)_{i \in S} \in \mathbb{R}^{|S|}$. The dot product of two vectors $u, v \in \mathbb{R}^n$ is $\langle u, v \rangle = \sum_{i \in [n]} u_i v_i \in \mathbb{R}$, and the element-wise product is $u \cdot v = (u_1 v_1, \dots, u_n v_n) \in \mathbb{R}^n$. If $u, v \in \{0, 1\}^n$ are two Boolean vectors, then we define the bitwise XOR of u and v as $u \oplus v = (u_1 \oplus v_1, \dots, u_n \oplus v_n) \in \{0, 1\}^n$. We consider the (partial) pointwise order over \mathbb{R}^n defined as $u \geq v$ if and only if $u_i \geq v_i$ for all $i \in [n]$ (we similarly define $u \leq v$, $u > v$, etc.). Finally, given a real number $\beta \neq 0$, we let β^u denote the vector $(\beta^{u_1}, \dots, \beta^{u_n})$.

2.2 Concentration inequalities

We describe several concentration inequalities for bounding the tail of a random variable X in terms of its expectation $\mathbb{E}[X]$ and its variance $\text{Var}[X]$. We refer the reader to [BLM13] for more details and references. First, we present the Markov inequality that gives an upper bound on the probability that a random variable is larger than some number.

Theorem 2.2.1 (MARKOV'S INEQUALITY). *Suppose X is a non-negative random variable. Then for any $a > 0$,*

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}.$$

The next four theorems quantify the deviation of a sum of random variables from its mean.

Theorem 2.2.2 (CHEBYSHEV'S INEQUALITY). *Suppose X_1, \dots, X_t are pairwise independent random variables with finite variance. Let $M_t = (X_1 + \dots + X_t)/t$ denote their average value. Then for any $\delta \in (0, 1)$,*

$$\Pr \left[|M_t - \mu| \geq \sqrt{\frac{\sigma^2}{t\delta}} \right] \leq \delta$$

where $\mu = \mathbb{E}[M_t]$ and $\sigma^2 = \frac{1}{t} \sum_{i=1}^t \text{Var}[X_i]$.

The Hoeffding inequality provides a better dependence on the failure probability δ , but it does not depend on the variance. Note that the same inequality holds with $\mu - M_t$ instead of $M_t - \mu$ by substituting $X_i \mapsto -X_i$.

Theorem 2.2.3 (Hoeffding's Inequality). *Suppose X_1, \dots, X_t are independent random variables such that $a \leq X_i \leq b$ for all i . Let $M_t = (X_1 + \dots + X_t)/t$ denote their average value. Then for any $\delta \in (0, 1)$,*

$$\Pr \left[M_t - \mu \geq \sqrt{\frac{(b-a)^2 \log(1/\delta)}{2t}} \right] \leq \delta$$

where $\mu = \mathbb{E}[M_t]$.

The next multiplicative Chernoff bound offers a better dependence on the mean for binary random variables and relative error approximation.

Theorem 2.2.4 (Chernoff's Bound). *Suppose X_1, \dots, X_t are independent random variables taking values in $\{0, 1\}$. Let $M_t = (X_1 + \dots + X_t)/t$ denote their average value. Then for any $0 < \epsilon < 1$,*

$$(\text{Multiplicative}) \quad \Pr[M_t - \mu \leq -\epsilon\mu] \leq \exp\left(-\frac{t\mu\epsilon^2}{2}\right) \quad \text{and} \quad \Pr[M_t - \mu \geq \epsilon\mu] \leq \exp\left(-\frac{t\mu\epsilon^2}{3}\right)$$

$$(\text{Additive}) \quad \Pr[M_t - \mu < -\epsilon] \leq \exp(-2t\epsilon^2) \quad \text{and} \quad \Pr[M_t - \mu > \epsilon] \leq \exp(-2t\epsilon^2)$$

where $\mu = \mathbb{E}[M_t]$.

The Chernoff bound is used in the “median trick” (also called the “powering lemma” [JV86]) to boost the success probability of an algorithm that outputs some correct real value with probability $\mu > 1/2$. The procedure consists of running several independent copies of the algorithm and taking the median of the obtained results. If we let X_i denote the binary random variable that equals 1 when the i -th run is correct, then the next result gives an upper bound on the number of repetitions needed to achieve a success probability of $1 - \delta$ for any $\delta \in (0, 1)$.

Corollary 2.2.5 (Median Trick). *Suppose X_1, \dots, X_t are independent random variables taking values in $\{0, 1\}$ where $\mathbb{E}[X_i] \geq 1/2 + \epsilon$ for all i and some $\epsilon > 0$. For any $\delta \in (0, 1)$, if $t \geq \frac{\log(1/\delta)}{2\epsilon^2}$ then $\text{median}(X_1, \dots, X_t) = 1$ with probability at least $1 - \delta$.*

Finally, the Bernstein inequality is a refinement of Hoeffding's inequality that introduces a dependence on the variance.

Theorem 2.2.6 (Bernstein's Inequality). *Suppose X_1, \dots, X_t are independent random variables such that $|X_i| \leq b$ for all i . Let $M_t = (X_1 + \dots + X_t)/t$ denote their average value. Then for any $t \geq 0$,*

$$\Pr \left[M_t - \mu \geq \sqrt{\frac{2\sigma^2 \log(1/\delta)}{t}} + \frac{3b \log(1/\delta)}{2t} \right] \leq \delta$$

where $\mu = \mathbb{E}[M_t]$ and $\sigma^2 = \frac{1}{t} \sum_{i=1}^t \text{Var}[X_i]$.

3

Algorithmic Preliminaries

3.1 Quantum circuit model

In this section, we present the standard model of computation used to describe a quantum algorithm. We highlight the main properties of this model that are needed later in the thesis. We refer the reader to the books [NC11; KLM07; LR14] and to the lecture notes [Wol19; ODo15; Chi17] for a more general introduction to quantum computing.

A *quantum circuit* [Deu89; Yao93] is a sequence of elementary quantum operations (the *quantum gates*), that operate in a predefined order on a collection of qubits (the *quantum memory*) represented as a *state vector* $|\psi\rangle \in \mathcal{H}$ in some Hilbert space \mathcal{H} . In the graphical representation of a quantum circuit, each qubit is depicted as a wire that passes through a series of gates, where the time axis is to be read from left to right. Each gate operates on the wires that are incident to it according to some predefined rule. The initial state of the memory is written on the left of the circuit. In most cases, each qubit starts in the state $|0\rangle$, or in a state $|x\rangle$ where $x \in \{0, 1\}$ encodes an input to the computation. An example of a quantum circuit is given in Figure 3.1. The computation performed by this circuit will become clear in the next paragraphs, where we describe the different types of quantum gates that compose it.

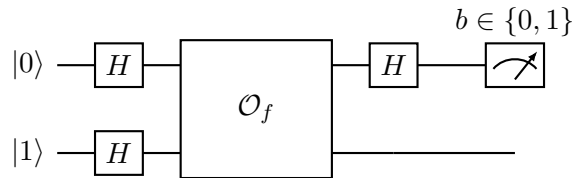
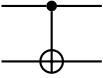


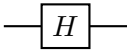
Figure 3.1: Deutsch’s algorithm that outputs $b = 0$ when $f : \{0, 1\} \rightarrow \{0, 1\}$ is constant.

Universal gate set. The postulates of quantum mechanics imply that the state vector of a closed quantum system must undergo a *unitary* transformation. Accordingly, a quantum gate is defined as a unitary transformation that operates on few qubits (one or two in general), and that can be composed with other gates to construct more complicated unitary transformations. As in the classical setting, where any computation can be performed using AND, OR and NOT gates, some sets of quantum gates are *universal* in the sense that they allow to represent any unitary transformation. One such example is the set made of the controlled negation gate CNOT (acting on two qubits) and of all 1-qubit gates. The CNOT gate is represented below, both in its graphical and matrix form (expressed in the standard basis). It negates the value of the second qubit when the first one is

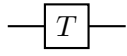
equal to $|1\rangle$. The set of all 1-qubit gates includes for instance the Hadamard transform H and the $\pi/8$ gate T (also represented below). For practical implementation purposes, the Solovay-Kitaev theorem guarantees that any 1- or 2-qubits gate can be approximated up to error ϵ (in spectral norm) by using only $\text{polylog}(1/\epsilon)$ gates from the set $\{\text{CNOT}, H, T\}$.



$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

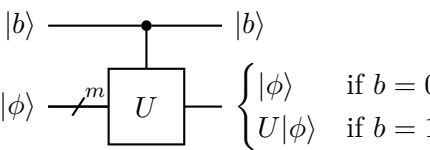


$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$



$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

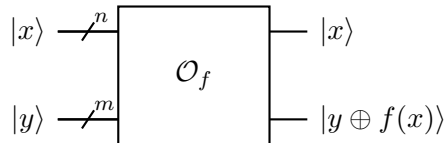
In this thesis, we make use of a universal gate set together with some ad-hoc quantum gates (such as the oracle gates defined in the next paragraph) depending on the problem under consideration. In particular, given a gate U acting on m qubits, we often use the *controlled gate* $\text{C}(U) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U$ acting on $m+1$ qubits that is described below. The strike symbol is a shortcut for m wires.



$$\begin{cases} |\phi\rangle & \text{if } b = 0 \\ U|\phi\rangle & \text{if } b = 1 \end{cases}$$

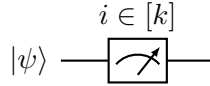
$$\text{C}(U) = \left(\begin{array}{c|c} I_{2^m} & 0 \\ \hline 0 & U \end{array} \right)$$

Oracle gate. In many quantum algorithms, the input to the computation is encoded as an *oracle* gate that provides coherent access to the data. An oracle gate is a *black-box* whose internal working is left unspecified. In practice, it may correspond to an auxiliary quantum circuit generated by a separate process, or it may represent the access to an external quantum memory. A first type of oracle gate is the *query* oracle, where given an input function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ the query gate \mathcal{O}_f (represented in the picture below) allows one to evaluate f in a reversible manner. The action of this gate on a computational basis state $|x\rangle|y\rangle$, where $x \in \{0, 1\}^n$ and $y \in \{0, 1\}^m$, is defined as $\mathcal{O}_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle$. We assume that the controlled gate $\text{C}(\mathcal{O}_f)$ is also available. This input model is extensively used in Chapters 6–8, 10. It can easily be extended to functions that are defined on other domains and ranges of values. We also give a variant of the query gate in Chapter 10, where the result of a query is encoded into the phase rather than in a separate quantum register.

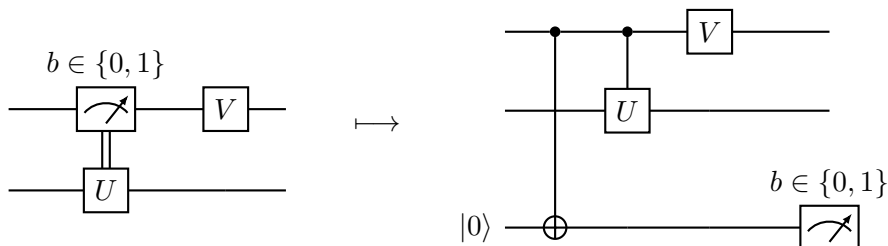


A second type of oracle gate, that generalizes the above approach, consists of having access to an unknown transformation U selected from a predefined set of unitaries. This transformation needs not be a permutation of the computational basis states, as was the case with the query gate \mathcal{O}_f defined before. In general, we assume that the inverse U^{-1} and the controlled versions $\text{C}(U)$ and $\text{C}(U^{-1})$ of U are also available as oracle gates. This model is used in Chapters 4, 5 and in the amplitude amplification and amplitude estimation algorithms presented in Sections 3.2, 3.3.

Measurement. A measurement provides a way to extract classical information from a closed quantum system. In this thesis, we use the notion of *projective measurement*, described by a family Π_1, \dots, Π_k of pairwise orthogonal projectors that sum to the identity. The effect of performing such a measurement on a state vector $|\psi\rangle$ is to observe the value $i \in [k]$ and to *collapse* the state vector to $\frac{1}{\|\Pi_i|\psi\rangle\|} \Pi_i|\psi\rangle$ with probability $\|\Pi_i|\psi\rangle\|^2$ (this is known as the Born rule). We augment the circuit model with a special gate, denoted by a meter symbol, that represents the use of a projective measurement (see the picture below, where the measurement outcome is written above the box). Unless otherwise stated, we use the *measurement in the computational basis* defined by the family $\{|i\rangle\langle i|\}_{0 \leq i < k}$ acting on a k -qubit state. The measurement operation requires modifying the quantum circuit formalism [AKN98] since it is not a unitary transformation. The state of the quantum memory, at any time of the computation, is now represented as a probability distribution over state vectors $\{p_\ell, |\psi_\ell\rangle\}_\ell$ (also known as a *mixed state*), meaning that the system is in the *pure state* $|\psi_\ell\rangle$ with probability p_ℓ . The mixed state evolves into $\{p_\ell, U|\psi_\ell\rangle\}_\ell$ after applying a unitary transformation U , and into $\{p_\ell \|\Pi_i|\psi_\ell\rangle\|^2, \frac{1}{\|\Pi_i|\psi_\ell\rangle\|} \Pi_i|\psi_\ell\rangle\}_{\ell,i}$ after performing a measurement $\{\Pi_i\}_i$.



Deferred measurement principle. A circuit that performs no measurement is called a *unitary circuit*. This property is often a prerequisite for the use of more advanced quantum algorithms that must run the inverse of the original circuit (see Sections 3.2, 3.3 and Chapters 4, 5). It also allows the application of certain lower bound methods (such as in Chapter 10) where the quantum memory is assumed to be represented as a single pure state. The *deferred measurement principle* (described in [AKN98, Lemma 4] and in [NC11, Section 4.4]), also called *principle of safe storage*, allows one to postpone all the intermediate measurements to the end of a quantum circuit without changing the outcome of the computation. The unitary part of the resulting circuit can then be used separately. The process of removing an intermediate measurement is illustrated in the figure below for the case of the single-qubit measurement $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$. The measured qubit is entangled with a new qubit, that is left unchanged afterward, by using a CNOT gate. The new qubit needs not be measured if its value is not part of the output. Each operation that is *classically* controlled on the measurement outcome (such as the unitary U in the left circuit below, which is applied if and only if $b = 1$) is replaced with a quantum controlled operation. The box V represents any subsequent computation in the example. In general, this technique requires increasing the memory size by one qubit each time a binary measurement is postponed. Thus, it must be used carefully when the memory size is a concern (as is the case in Part IV).



Simulation of classical computation. The simulation of a classical circuit by a quantum one requires to make the original circuit *reversible* (such that it can be extended by linearity to a unitary transformation). The canonical way of doing that is to replace each classical gate $g : \{0, 1\}^a \rightarrow \{0, 1\}^b$ with the reversible gate $\mathcal{R}_g : \{0, 1\}^{a+b} \rightarrow \{0, 1\}^{a+b}$ defined as $\mathcal{R}_g(x, y) = (x, y \oplus g(x))$. This process often increases the memory size of the circuit. Bennett [Ben73] showed that any classical circuit that uses T gates and S bits of memory can be turned into a reversible circuit that uses $O(T)$ gates and $O(T + S)$ bits of memory. There exist other simulation techniques [Ben89; LS90; LMT00] that can lower the memory size of the resulting reversible circuit at the cost of using more gates. For instance, Bennett [Ben89] described a different construction that uses $O(T^2)$ gates and only $O(S \log T)$ bits of memory.

Measures of complexity. There exist several ways of measuring the performances of a quantum circuit. We mention three measures that are used throughout this thesis. The *gate* complexity of a circuit is defined as its total number of elementary gates. The *query* complexity counts only the query gates. These two measures are often used to characterize the computation time of an algorithm. The *space* complexity is defined as the number of qubits on which the circuit is operating.

Circuit notations. If \mathcal{C} is a unitary circuit then we represent the corresponding unitary transformation with the symbol \mathcal{C} as well. In particular, $\mathcal{C}|\psi\rangle$ is the state vector obtained by running \mathcal{C} on an initial state vector $|\psi\rangle$, and \mathcal{C}^{-1} is the inverse unitary transformation. The transformation \mathcal{C}^{-1} can be implemented by running the original circuit \mathcal{C} backward, where each quantum gate is replaced with its inverse. Given two circuits \mathcal{C} and \mathcal{C}' operating on the same quantum register, we let $\mathcal{C} \parallel \mathcal{C}'$ denote the concatenated circuit that runs \mathcal{C} first and then \mathcal{C}' .

3.2 Amplitude amplification

The amplitude amplification algorithm [BHMT02] is a generalization of Grover's quantum search [Gro96a] to the problem of boosting the success probability of a unitary quantum algorithm, quadratically faster than it is possible classically. The main property of this algorithm is given below. This result corresponds to Equation (8) in [BHMT02].

Theorem 3.2.1 (AMPLITUDE AMPLIFICATION, [BHMT02]). *Let U be a unitary quantum algorithm and let Π be a projection operator. Consider the angle $\theta \in [0, \frac{\pi}{2}]$ and two unit states $|\psi_0\rangle, |\psi_1\rangle$ such that $\sin(\theta)|\psi_1\rangle = \Pi U|\mathbf{0}\rangle$ and $U|\mathbf{0}\rangle = \cos(\theta)|\psi_0\rangle + \sin(\theta)|\psi_1\rangle$. Then, for any integer $t \geq 0$, the amplitude amplification algorithm implements a unitary transformation $\text{AAmp}(U, \Pi, t)$ that satisfies*

$$\text{AAmp}(U, \Pi, t)|\mathbf{0}\rangle = \cos((2t + 1)\theta)|\psi_0\rangle + \sin((2t + 1)\theta)|\psi_1\rangle.$$

The algorithm uses $t + 1$ applications of U , t applications of U^\dagger , and t applications of the reflection operator $I - 2\Pi$.

If we let $\sqrt{p} = \sin(\theta)$ denote the amplitude of the state $|\psi_1\rangle$ in $U|\mathbf{0}\rangle$, then the amplitude amplification algorithm requires roughly $1/\sqrt{p}$ iterations to increase the amplitude of $|\psi_1\rangle$ to a constant number (whereas $\Omega(1/\sqrt{p})$ repetitions of the original algorithm would be necessary classically). We have the following quantitative result (used in Proposition 5.3.2).

Corollary 3.2.2 (Lemma 5.2 in [AA05]). *Under the hypothesis and notations of Theorem 3.2.1, let $p = \sin^2(\theta)$ and $p' = \sin^2((2t+1)\theta)$ denote the squared amplitude of $|\psi_1\rangle$ in $U|\mathbf{0}\rangle$ and $\text{AAmp}(U, \Pi, t)|\mathbf{0}\rangle$ respectively. If $t \leq \frac{\pi}{4\arcsin\sqrt{p}} - \frac{1}{2}$ then p' satisfies $p' \geq \left(1 - \frac{(2t+1)^2}{3}p\right)(2t+1)^2p$.*

Next, we present a variant of the amplitude amplification algorithm that does not use a pre-defined number of computation steps. We call it the “sequential amplitude amplification” algorithm in reference to *sequential analysis*. The original version of this algorithm was analyzed in Theorem 3 of [BBHT98; BHMT02], with a bound on the expected running time $\mathbb{E}[T]$. We complement this result with a lower tail bound on T . In order to simplify the analysis, we propose a slightly different version of the algorithm, where a parameter t is sampled in an interval $[\lambda^{\ell-1}, \lambda^\ell - 1]$ instead of $[0, \lambda^\ell - 1]$. The next theorem is used in Lemma 4.3.2 and Proposition 7.4.2.

1. Set $\ell = 0$ and $\lambda = 6/5$.
2. Increase ℓ by 1 and choose an integer $t \in [\lambda^{\ell-1}, \lambda^\ell - 1]$ uniformly at random.
3. Apply the **amplitude amplification** algorithm $\text{AAmp}(U, \Pi, t)$ to $|\mathbf{0}\rangle$ and measure the state by using the projective measurement $\{I - \Pi, \Pi\}$. If the outcome is “ Π ” then stop and output the obtained state. Otherwise, go to step 2.

Algorithm 3.2: Sequential amplitude amplification, $\text{Seq-AAmp}(U, \Pi)$.

Theorem 3.2.3 (SEQUENTIAL AMPLITUDE AMPLIFICATION). *Let U be a unitary quantum algorithm and let Π be a projection operator. Consider the number $p \in [0, 1]$ and two unit states $|\psi_0\rangle, |\psi_1\rangle$ such that $\sqrt{p}|\psi_1\rangle = \Pi U|\mathbf{0}\rangle$ and $U|\mathbf{0}\rangle = \sqrt{1-p}|\psi_0\rangle + \sqrt{p}|\psi_1\rangle$. If $p > 0$ then the sequential amplitude amplification algorithm $\text{Seq-AAmp}(U, \Pi)$ (Algorithm 3.2) outputs the state $|\psi_1\rangle$ with probability 1. Moreover, if we let T denote the number of applications of U , U^\dagger and $I - 2\Pi$ used by the algorithm, then*

(1) $\mathbb{E}[T] \leq O(1/\sqrt{p})$.

(2) *There is a universal constant c such that $\Pr[T < c/\sqrt{p}] \leq 1/10$.*

Proof. Let $0 \leq \theta \leq \pi/2$ be the angle such that $\sqrt{p} = \sin \theta$. We show the theorem in the case where $\theta < \pi/4$ (the case $\theta \geq \pi/4$ is easy to handle separately). Let P_ℓ denote the probability of obtaining “ Π ” (i.e. the state $|\psi_1\rangle$ is returned) at step 3 of the ℓ -th iteration.

We first prove part (1). Let $\ell^+ = \left\lceil \log_\lambda \left(\frac{\lambda}{(\lambda-1)\sin(2\theta)} \right) \right\rceil$. If $\ell \geq \ell^+$ then we have $P_\ell = \frac{1}{(\lambda-1)\lambda^{\ell-1}} \sum_{t=\lambda^{\ell-1}}^{\lambda^\ell-1} \sin^2((2t+1)\theta) \geq \frac{1}{2} - \frac{1}{4(\lambda-1)\lambda^{\ell-1}\sin(2\theta)} \geq \frac{1}{4}$ where the equality is by Theorem 3.2.1, the first inequality uses the same trigonometric identities as in the proof of [BBHT98, Lemma 2], and the second one uses that $\ell \geq \ell^+$. Moreover, the algorithm has used at most $\sum_{t=1}^{\ell} \lambda^t \leq 5\lambda^{\ell+1}$ applications of U , U^\dagger and $I - 2\Pi$ after ℓ iterations of step 3. Consequently, $\mathbb{E}[T] \leq \sum_{\ell \geq \ell^+} 5\lambda^{\ell+1}(3/4)^{\ell-\ell^+} \leq O(\lambda^{\ell^+}) \leq O(1/\sqrt{p})$.

We prove part (2). Let $\ell^- = \lfloor \log_\lambda(\frac{1}{12\theta}) \rfloor$. If $\ell \leq \ell^-$ then $P_\ell \leq \sin^2((2(\lambda^\ell - 1) + 1)\theta) \leq 4\lambda^{2\ell}\theta^2 \leq \frac{1}{36}\lambda^{2(\ell-\ell^-)}$, where the first inequality is by Theorem 3.2.1, and the second one uses that $\sin(x) \leq x$ for all $x \in [0, \pi/2]$. Thus, the probability that the algorithm stops before the ℓ^- -th iteration is at most $\sum_{\ell=1}^{\ell^-} \frac{1}{36}\lambda^{2(\ell-\ell^-)} \leq 1/10$. Moreover, the algorithm has used at least $\lambda^{\ell^- - 1} \geq \Omega(1/\sqrt{p})$ applications of U , U^\dagger and $I - 2\Pi$ after ℓ^- iterations. \square

3.3 Amplitude estimation

The amplitude estimation algorithm [BHMT02] is a generalization of quantum counting [BBHT98] to the problem of estimating the success probability of an algorithm. The next result corresponds to Theorems 11 and 12 in [BHMT02].

Theorem 3.3.1 (AMPLITUDE ESTIMATION, [BHMT02]). *Let U be a unitary quantum algorithm and let Π be a projection operator. Define the number $p \in [0, 1]$ such that $p = \|\Pi U|0\rangle\|^2$. Then, for any integer $t \geq 0$, the amplitude estimation algorithm $\text{AEst}(U, \Pi, t)$ outputs an amplitude estimate \tilde{p} such that,*

$$\Pr \left[|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{t} + \frac{\pi^2}{t^2} \right] \geq 8/\pi^2.$$

If we let $\theta \in [0, \pi/2]$ denote the angle such that $\sqrt{p} = \sin \theta$, then $\Pr[\tilde{p} = 0] = \frac{\sin^2(t\theta)}{t^2 \sin^2(\theta)}$. The algorithm uses t applications of U , U^\dagger , $I - 2\Pi$ and $O(\log^2(t))$ other 2-qubit quantum gates.

The next corollary gives an upper bound for the probability that the estimate \tilde{p} is of an order larger than p . This is similar in some sense to Markov's inequality.

Corollary 3.3.2. *Under the hypothesis and notations of Theorem 3.3.1, the output \tilde{p} of the amplitude estimation algorithm satisfies $\tilde{p} \leq (1 + 2\pi)^2 p$ with probability at least $8/\pi^2$.*

Proof. If $t \geq 1/(2\sqrt{p})$ then Theorem 3.2.1 implies that $\tilde{p} \leq p + 4\pi p + 4\pi^2 p^2 \leq (1 + 2\pi)^2 p$ with probability at least $8/\pi^2$. If $t < 1/(2\sqrt{p})$ then let $\theta \in [0, \pi/2]$ denote the angle such that $\sqrt{p} = \sin \theta$. Observe that $\theta \leq \frac{\pi}{2}\sqrt{p} \leq \frac{\pi}{4t}$ (using the standard inequality $\frac{2}{\pi}\theta \leq \sin(\theta)$) and $\frac{\sin^2(t\theta)}{t^2 \sin^2(\theta)} \geq \frac{\sin^2(t\pi/(4t))}{t^2 \sin^2(\pi/(4t))} \geq \frac{\sin^2(\pi/4)}{t^2 (\pi/(4t))^2} = 8/\pi^2$, since $x \mapsto \sin^2(tx)/(t^2 \sin^2(x))$ is decreasing for $0 < x \leq \pi/t$. Thus, \tilde{p} is equal to 0 with probability at least $8/\pi^2$ when $t < 1/(2\sqrt{p})$, according to Theorem 3.2.1. \square

We present a sequential version of the amplitude estimation algorithm that does not need a time parameter t as input. This result was first obtained by [BHMT02, Theorem 15]. We describe a variant with two main additional properties: a bound on the expected value of the estimate and of its inverse (part (2)) and a high-probability bound on the running time (part (4)). These results are used in Proposition 4.5.1 and Proposition 5.3.4.

Theorem 3.3.3 (SEQUENTIAL AMPLITUDE ESTIMATION). *Let U be a unitary quantum algorithm and let Π be a projection operator. Define the number $p \in [0, 1]$ such that $p = \|\Pi U|0\rangle\|^2$. Fix two reals $\epsilon, \delta \in (0, 1/2)$. Then, the sequential amplitude estimation algorithm $\text{Seq-AEst}(U, \Pi, \epsilon, \delta)$ (Algorithm 3.3) outputs an amplitude estimate \tilde{p} and uses a number T of applications of U , U^\dagger , $I - 2\Pi$ such that,*

- (1) $\Pr[|\tilde{p} - p| > \epsilon p] \leq \delta$.
- (2) $\mathbb{E}[1/\tilde{p}] \leq O(1/p)$ and $\mathbb{E}[\sqrt{\tilde{p}}] \leq O(\sqrt{p})$.
- (3) $\mathbb{E}[T] \leq O\left(\frac{\log(1/\delta)}{\epsilon\sqrt{p}}\right)$.
- (4) There is a universal constant c such that $\Pr\left[T > c \frac{\log(1/\delta)}{\epsilon\sqrt{p}}\right] \leq \delta$.

Moreover, it uses $O(\log^2(T))$ other 2-qubit quantum gates.

1. Set $\ell = 0$ and $\lambda = 6/5$.
2. Increase ℓ by 1.
 - a) For $i = 1, \dots, 24\lceil\log(5/\delta)\rceil$: choose an integer $t_i \in [\lambda^{\ell-1}, \lambda^\ell - 1]$ uniformly at random, apply the **amplitude amplification** algorithm $\text{AAmp}(U, \Pi, t_i)$ to $|0\rangle$ and measure the state by using the projective measurement $\{I - \Pi, \Pi\}$. If the outcome is “ Π ” then set $b_i^{(\ell)} = 1$, else set $b_i^{(\ell)} = 0$.
 - b) Let $b^{(\ell)} = \frac{\sum_{i=1}^{24\lceil\log(5/\delta)\rceil} b_i^{(\ell)}}{24\lceil\log(5/\delta)\rceil}$. If $b^{(\ell)} < 1/12$ then go to step 2, else set $q = \lambda^{-2\ell}$.
3. For $j = 1, \dots, 6\lceil\log(2/\delta)\rceil$: compute an estimate \tilde{p}_j by using the **amplitude estimation** algorithm $\text{AEst}(U, \Pi, \lceil\frac{400}{\epsilon\sqrt{q}}\rceil)$. Set $q' = \text{median}(\tilde{p}_1, \dots, \tilde{p}_{6\lceil\log(2/\delta)\rceil})$.
4. Output $\tilde{p} = \text{median}(2^{-12}q, q', 9q)$.

Algorithm 3.3: Sequential amplitude estimation, $\text{Seq-AEst}(U, \Pi, \epsilon, \delta)$.

Proof. Let $0 \leq \theta \leq \pi/2$ be the angle such that $\sqrt{p} = \sin \theta$ and define $\ell^- = \lfloor \log_\lambda(\frac{1}{12\theta}) \rfloor$ and $\ell^+ = \lceil \log_\lambda(\frac{\lambda}{(\lambda-1)\sin(2\theta)}) \rceil$ (assuming $\theta < \pi/4$, the case $\theta \geq \pi/4$ is easy to handle separately). Let P_ℓ denote the probability of obtaining $b_i^{(\ell)} = 1$ at step 2.a. We have shown in the proof of Theorem 3.2.3 that $\Pr[b_i^{(\ell)} = 1] \geq 1/4$ when $\ell \geq \ell^+$, and $\Pr[b_i^{(\ell)} = 1] \leq \frac{1}{36}\lambda^{2(\ell-\ell^-)}$ when $\ell \leq \ell^-$. We show that the average $b^{(\ell)}$ computed at step 2.b satisfies

$$(i) \Pr[b^{(\ell)} < 1/12] \leq \delta/5 \text{ if } \ell \geq \ell^+, \quad (ii) \Pr[b^{(\ell)} \geq 1/12] \leq (\delta/5)^{\ell^- - \ell} \text{ if } \ell \leq \ell^-.$$

Part (i) is obtained by the additive Chernoff bound. Part (ii) is obtained by $\Pr[b^{(\ell)} \geq 1/12] \leq \binom{24\lceil\log(5/\delta)\rceil}{2\lceil\log(5/\delta)\rceil} P_\ell^{2\lceil\log(5/\delta)\rceil} \leq (\frac{12e}{36}\lambda^{2(\ell-\ell^-)})^{2\lceil\log(5/\delta)\rceil} \leq (\delta/5)^{\ell^- - \ell}$, where we used that $\binom{n}{k} \leq (en/k)^k$ for all $k \leq n$, and $\lambda^4 > 2$.

We now prove that the estimate q obtained at step 2.b satisfies the three properties:

$$(a) \Pr[p/6 \leq q \leq 2^{11}p] \geq 1 - \delta/2, \quad (b) \mathbb{E}[1/q] \leq O(1/p), \quad (c) \mathbb{E}[\sqrt{q}] \leq O(\sqrt{p}).$$

The algorithm sets $q = \lambda^{-2\ell}$ when $b^{(\ell)} \geq 1/12$. Consequently, by (i) and (ii), we have $\Pr[q > \lambda^{-2\ell^-}] \leq \sum_{\ell \leq \ell^-} (\delta/5)^{\ell^- - \ell} \leq \delta/4$ and $\Pr[q < \lambda^{-2\ell^+}] \leq \delta/5$. Moreover, $\lambda^{-2\ell^+} \geq \frac{(\lambda-1)^2}{\lambda^4} \sin^2(2\theta) \geq p/6$, and $\lambda^{-2\ell^-} \leq (12\lambda\theta)^2 \leq 2^{11}p$ since $x \leq (\pi/2)\sin(x)$ when $x \in [0, \pi/2]$. This proves part (a). Parts (b) and (c) are obtained by $\mathbb{E}[1/q] \leq \sum_{\ell \geq \ell^+} \lambda^{2\ell} (\delta/5)^{\ell - \ell^+} \leq O(\lambda^{2\ell^+})$ and $\mathbb{E}[\sqrt{q}] \leq \sum_{\ell \leq \ell^-} \lambda^{-\ell} (\delta/5)^{\ell^- - \ell} \leq O(\lambda^{-\ell^-})$.

We finally prove the different parts of the theorem. If $q \leq 2^{11}p$ then $\Pr[|\tilde{p}_j - p| \leq \epsilon p] \geq 8/\pi^2$ for each j at step 3 according to Theorem 3.3.1. In this case, by the median trick, we have $\Pr[|q' - p| > \epsilon p] \leq \delta/2$. Part (1) is deduced from $\Pr[|\tilde{p} - p| \leq \epsilon p] \geq \Pr[|q' - p| \leq \epsilon p \text{ and } 2^{-12}q \leq q' \leq 9q] \geq \Pr[|q' - p| \leq \epsilon p \text{ and } p/6 \leq q \leq 2^{11}p] \geq 1 - \delta$, where we used (a) for the last inequality. Part (2) is deduced from (b), (c) and the fact that $2^{-12}q \leq \tilde{p} \leq 9q$ (by definition of step 4). Finally, parts (3) and (4) are deduced from (a), (b) and the fact that $T = O\left(\frac{\log(1/\delta)}{\epsilon\sqrt{q}}\right)$. \square

Part II

Quantum Algorithms for Estimating Average Values

4

Mean Estimation Problem

This chapter is based on the following papers:

[HM19] Y. Hamoudi and F. Magniez. “Quantum Chebyshev’s Inequality and Applications”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 69:1–69:16.

[Ham21] Y. Hamoudi. “Quantum Sub-Gaussian Mean Estimator”. In submission. 2021.

4.1 Introduction

The problem of estimating the mean μ of a real-valued random variable X given *i.i.d.* samples from it is one of the most basic tasks in statistics and in the Monte Carlo method. The properties of the various classical *mean estimators* are well understood. The standard non-asymptotic criterion used to assess the quality of an estimator is formulated as the following *high probability deviation bound*: upon performing t random experiments that return t samples from X , and given a failure probability $\delta \in (0, 1)$, what is the smallest error $\epsilon(t, \delta, X)$ such that the output $\tilde{\mu}$ of the estimator satisfies $|\tilde{\mu} - \mu| > \epsilon(t, \delta, X)$ with probability at most δ ? Under the standard assumption that the unknown random variable X has a finite variance σ^2 , the best possible performances are obtained by the so-called *sub-Gaussian estimators* [LM19] that achieve the following deviation bound

$$\Pr \left[|\tilde{\mu} - \mu| > L \sqrt{\frac{\sigma^2 \log(1/\delta)}{t}} \right] \leq \delta \quad (4.1)$$

for some constant L . The term “sub-Gaussian” reflects that these estimators have a Gaussian tail even for non-Gaussian distributions. The most well-known sub-Gaussian estimator is arguably the *median-of-means* [NY83; JVV86; AMS99], which consists of partitioning the t samples into roughly $\log(1/\delta)$ groups of equal size, computing the empirical mean over each group, and returning the median of the obtained means.

The process of generating a random sample from X is generalized in the quantum model by assuming the existence of a unitary operator U where $U|0\rangle$ coherently encodes the distribution of X . A *quantum experiment* is then defined as one application of this operator or its inverse. The celebrated quantum amplitude estimation algorithm [BHMT02] provides a way to estimate the mean of any *Bernoulli* random variable by performing fewer experiments than with any classical estimator. Yet, for general distributions, the existing quantum mean estimators either require additional information on the variance [Hei02; Mon15; HM19] or are less performant than the classical sub-Gaussian estimators when the distribution is heavy tailed [BHMT02; Ter99; BDGT11; Mon15]. These results leave open the existence of a general quantum speedup for the mean estimation problem. We

address this question by introducing the concept of *quantum sub-Gaussian estimators*, defined through the following deviation bound

$$\Pr\left[|\tilde{\mu} - \mu| > L \frac{\sigma \log(1/\delta)}{t}\right] \leq \delta \quad (4.2)$$

for some constant L . We give the first construction of a quantum estimator that achieves this bound up to a logarithmic factor in t . Additionally, we prove that it is impossible to go below that deviation level. This result provides a clear equivalent of the concept of sub-Gaussian estimator in the quantum setting.

A second important family of mean estimators addresses the (ϵ, δ) -approximation problem, where given a fixed relative error $\epsilon \in (0, 1)$ and a failure probability $\delta \in (0, 1)$ the goal is to output a mean estimate $\tilde{\mu}$ such that

$$\Pr[|\tilde{\mu} - \mu| > \epsilon|\mu|] \leq \delta. \quad (4.3)$$

The aforementioned sub-Gaussian estimators do not quite answer this question since the number of experiments they require (respectively $t = \Omega((\frac{\sigma}{\epsilon\mu})^2 \log(1/\delta))$ and $t = \tilde{\Omega}(\frac{\sigma}{\epsilon|\mu|} \log(1/\delta))$) depends on the *unknown* quantities σ and μ . Sometimes a good upper bound is known on the *coefficient of variation* $|\sigma/\mu|$ and can be used to parametrize a sub-Gaussian estimator. Otherwise, the standard approach is based on *sequential analysis* techniques, where the number of experiments is chosen adaptively depending on the results of previous computations. Given a random variable distributed in $[0, 1]$, the optimal classical estimators perform $\Theta((\frac{\sigma}{\epsilon\mu})^2 + \frac{1}{\epsilon\mu}) \log(1/\delta)$ random experiments *in expectation* [DKLR00] for computing an (ϵ, δ) -approximation of μ . We construct a quantum estimator that reduces this number to $\tilde{\Theta}((\frac{\sigma}{\epsilon\mu} + \frac{1}{\sqrt{\epsilon\mu}}) \log(1/\delta))$ and we prove that it is optimal. We also consider the situation where a non-increasing function f is known such that $f(\mu) \geq \sigma/\mu$. In that case, we present an estimator that performs roughly $\tilde{O}(\frac{f(c\mu)}{\epsilon} \log(1/\delta))$ quantum experiments with high probability for some constant c .

4.1.1 Related work

There is an extensive literature on classical sub-Gaussian estimators and we refer the reader to [LM19; Cat12; BCL13; DLLO16; LV20] for an overview of the main results and recent improvements. We point out that the *empirical mean* estimator is not sub-Gaussian, although it is optimal for Gaussian random variables [SV05; Cat12]. The non-asymptotic performances of the empirical mean estimator are captured by several standard concentration bounds such as those presented in Section 2.2.

There is a series of quantum mean estimators [Gro98; AW99; BDGT11] that get close to the bound $\Pr[|\tilde{\mu} - \mu| > L \frac{\log(1/\delta)}{t}] \leq \delta$ for any random variable distributed in $[0, 1]$ and some constant L . Similar results hold for numerical integration problems [AW99; Nov01; Hei02; TW02; Hei03]. The amplitude estimation algorithm [BHMT02; Ter99] leads to a sharper bound of $\Pr[|\tilde{\mu} - \mu| > L(\frac{\sqrt{\mu(1-\mu)} \log(1/\delta)}{t} + \frac{\log(1/\delta)^2}{t^2})] \leq \delta$ (see Proposition 4.4.1) when X is distributed in $[0, 1]$. Nevertheless, the quantity $\mu(1-\mu)$ is always larger than or equal to the variance σ^2 . The question of improving the dependence on σ^2 was considered in [Hei02; Mon15; HM19]. The estimators of [Hei02; Mon15] require us to know an upper bound Σ on the standard deviation σ , whereas [HM19] uses an upper bound \mathcal{V} on the coefficient of variation σ/μ (for non-negative random variables). The performances of these estimators are captured (up to logarithmic factors) by the deviation bound given in Equation (4.2) with σ replaced by Σ and $\mu\mathcal{V}$ respectively.

The (ϵ, δ) -approximation problem has been addressed by several classical works such as [DKLR00; MSA08; GNP13; Hub19]. In the quantum setting, there is a variant [BHMT02, Theorem 15] of the amplitude estimation algorithm that performs $O(\log(1/\delta)/(\epsilon\sqrt{\mu}))$ experiments in expectation and computes an (ϵ, δ) -approximate of the mean of any random variables distributed in $[0, 1]$ (see Theorem 3.3.3 and Proposition 4.5.1). However, the complexity of this estimator does not scale with σ . Given an upper bound \mathcal{V} on σ/μ , the estimator of [HM19] can be used to compute an (ϵ, δ) -approximate with roughly $\tilde{O}(\mathcal{V} \log(1/\delta)/\epsilon)$ quantum experiments.

We note that the related problem of estimating the mean with *additive* error ϵ , that is $\Pr[|\tilde{\mu} - \mu| > \epsilon] \leq \delta$, has also been considered by several authors. The optimal number of experiments is $\Theta(\log(1/\delta)/\epsilon^2)$ classically [CEG95] and $\Theta(1/\epsilon)$ quantumly [NW99] (with failure probability $\delta = 1/3$). These bounds do not depend on unknown parameters (as opposed to the relative error case), thus sequential analysis techniques are unnecessary here. Montanaro [Mon15] also described an estimator that performs $\tilde{O}(\Sigma \log(1/\delta)/\epsilon)$ quantum experiments given an upper bound Σ on the standard deviation σ .

4.1.2 Contributions and organization

We first formally define the input model in Section 4.2. We introduce the concept of “q-random variable” (Definition 4.2.3) to describe a random variable that corresponds to the output of a quantum computation. We measure the complexity of an algorithm by counting the number of *quantum experiments* (Definition 4.2.4) it performs with respect to a q-random variable.

We construct a quantum algorithm for estimating the quantiles of a q-random variable in Section 4.3, and we use it in Section 4.4 to design the following quantum sub-Gaussian estimator.

Theorem 4.4.2 (Restated). *There exists a quantum algorithm with the following properties. Let X be a q-random variable with mean μ and variance σ^2 , and set as input a time parameter t and a real $\delta \in (0, 1)$ such that $t \geq \log(1/\delta)$. Then, the algorithm outputs a mean estimate $\tilde{\mu}$ such that, $\Pr\left[|\tilde{\mu} - \mu| > \frac{\sigma \log(1/\delta)}{t}\right] \leq \delta$, and it performs $O(t \log^{3/2}(t) \log \log(t))$ quantum experiments. Moreover, if X is non-negative then $\Pr[\tilde{\mu} \leq (2 + 2\pi)^2 \mu] \geq 1 - \delta$.*

We prove in Section 4.6.1 that the above estimator is nearly optimal (Theorem 4.6.2). Then we turn our attention to the (ϵ, δ) -approximation problem in Section 4.5. We first construct an estimator that requires no prior information about the input random variable, except that it is distributed in $[0, 1]$. The number of experiments performed by this estimator is chosen *adaptively*, and the bound we get is stated in expectation.

Theorem 4.5.2 (Restated). *There exists a quantum algorithm with the following properties. Let X be a q-random variable distributed in $[0, 1]$ with mean μ and variance σ^2 , and set as input two reals $\epsilon, \delta \in (0, 1)$. Then, the algorithm outputs a mean estimate $\tilde{\mu}$ such that $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$, and it performs $\tilde{O}\left(\left(\frac{\sigma}{\epsilon\mu} + \frac{1}{\sqrt{\epsilon\mu}}\right) \log(1/\delta)\right)$ quantum experiments in expectation.*

We prove a nearly matching lower bound in Proposition 4.6.4. We also consider the (ϵ, δ) -approximation problem when some information is available on the coefficient of variation $|\sigma/\mu|$. The objective is to remove the above dependency on $1/\sqrt{\epsilon\mu}$. We use our sub-Gaussian estimator to simplify two results previously shown in [HM19]. First, we assume the knowledge of an upper bound \mathcal{V} on $|\sigma/\mu|$ and we obtain the next result that follows directly from Theorem 4.4.2.

Corollary 4.5.3 (Restated). *There exists a quantum algorithm with the following properties. Let X be a q -random variable with mean μ and variance σ^2 , and set as input a value $\mathcal{V} \geq |\sigma/\mu|$ and two reals $\epsilon, \delta \in (0, 1)$. Then, the algorithm outputs a mean estimate $\tilde{\mu}$ such that $\Pr[|\tilde{\mu} - \mu| > \epsilon|\mu|] \leq \delta$, and it performs $\tilde{O}(\frac{\mathcal{V}}{\epsilon} \log(1/\delta))$ quantum experiments.*

Then, we assume the knowledge of a non-increasing function f such that $f(\mu) \geq \sigma/\mu$. Although $f(\mu)$ is unknown *a priori*, we present an algorithm to approximate it with high success probability (Theorem 4.5.5). As a result, we obtain a variant of the above corollary where \mathcal{V} is replaced with a term on the order of $f(\mu)$ (Corollary 4.5.6).

Finally, we prove several lower bounds for the mean estimation problem in Section 4.6. In particular, we consider the weaker input model where one is given copies of a quantum state encoding the distribution of X . We prove that no quantum speedup is achievable in this setting (Theorem 4.6.6).

4.1.3 Proof overview

Sub-Gaussian estimator. Our approach (Theorem 4.4.2) combines several ideas used in previous classical and quantum mean estimators. In this section, we simplify the exposition by assuming that the random variable X is non-negative and by replacing the variance σ^2 with the second moment $\mathbb{E}[X^2]$. We also take the failure probability δ to be a small constant. Our starting point is a variant of the *truncated mean estimators* [Bic65; BCL13; LM19]. Truncation is a process that consists of replacing the samples larger than some threshold value with a smaller number. This has the effect of reducing the tail of the distribution, but also of changing its expectation. Here we study the effect of replacing the values larger than some threshold b with 0, which corresponds to the new random variable $Y = X\mathbb{1}_{X \leq b}$. We consider the following classical sub-Gaussian estimator that we were not able to find in the literature: set $b = \sqrt{t\mathbb{E}[X^2]}$ and compute the empirical mean of t samples from Y . By a simple calculation, one can prove that the expectation of the removed part is at most $\mathbb{E}[X - Y] \leq \mathbb{E}[X^2]/b = \sqrt{\mathbb{E}[X^2]}/t$. Moreover, using Bernstein's inequality and the boundedness of Y , the error between the output estimate and $\mathbb{E}[Y]$ is on the order of $\sqrt{\mathbb{E}[X^2]}/t$. These two facts together imply that the overall error for estimating $\mathbb{E}[X]$ is indeed of a sub-Gaussian type. This approach can be carried out in the quantum model by performing the truncation in superposition. This is similar to what is done in previous quantum mean estimators [Hei02; Mon15; HM19]. In order to obtain a quantum speedup, one must balance the truncation level differently by taking $b = t\sqrt{\mathbb{E}[X^2]}$. Then, by a clever use of amplitude estimation discovered by Heinrich [Hei02], the expectation of Y can be estimated with an error on the order of $\sqrt{\mathbb{E}[X^2]}/t$. The main drawback of this estimator is that it requires the knowledge of $\mathbb{E}[X^2]$ to perform the truncation. In previous work [Hei02; Mon15; HM19], the authors made further assumptions on the variance to be able to approximate b . Here, we overcome this issue by choosing the truncation level b differently. Borrowing ideas from classical estimators [LM19], we define b as the quantile value that satisfies $\Pr[X \geq b] = 1/t^2$. This quantile is always smaller than the previous threshold value $t\sqrt{\mathbb{E}[X^2]}$. Moreover, it can be shown that the removed part $\mathbb{E}[X - Y]$ is still on the order of $\sqrt{\mathbb{E}[X^2]}/t$. We give a new quantum algorithm for approximating this quantile with roughly t quantum experiments (Theorem 4.3.4), whereas it would require t^2 random experiments classically. Our quantile estimation algorithm builds upon the quantum minimum finding algorithm of Dürr and Høyer [DH96; AGGW20b] and the k th-smallest element finding algorithm of Nayak and Wu [NW99]. Importantly, it does not require any knowledge about $\mathbb{E}[X^2]$.

(ϵ, δ) -Approximation without side information. We follow an approach similar to that of a classical estimator described in [DKLR00]. Our algorithm (Theorem 4.5.2) uses the quantum sub-Gaussian estimator and the quantum *sequential Bernoulli estimator* described in Proposition 4.5.1. The latter estimator can estimate the mean μ of a random variable X distributed in $[0, 1]$ with constant relative error by performing $O(1/\sqrt{\mu})$ quantum experiments in expectation. The first step of the (ϵ, δ) -approximation algorithm is to compute a rough estimate m of μ with the sequential Bernoulli estimator. Then, the variance σ^2 of X is estimated by using again the sequential Bernoulli estimator on the random variable $(X - X')/2$ (where X' is an independent copy of X). The latter estimation is stopped if it uses more than $O(1/\sqrt{\epsilon m})$ quantum experiments. We show that if $\sigma^2 \geq \Omega(\epsilon\mu)$ then the computation is not stopped and the resulting estimate $\tilde{\sigma}^2$ is close to σ^2 with high probability. Otherwise, it is stopped with high probability and we set $\tilde{\sigma} = 0$. Finally, the quantum sub-Gaussian estimator is used with the parameter $t \approx \max\left(\frac{\tilde{\sigma}}{\epsilon m}, \frac{1}{\sqrt{\epsilon m}}\right)$ to obtain a refined estimate $\tilde{\mu}$ of μ . The choice of the first (resp. second) term in the maximum value implies that $|\tilde{\mu} - \mu| \leq \epsilon\mu$ with high probability when the variance σ^2 is larger (resp. smaller) than $\epsilon\mu$. In order to upper bound the expected number of experiments performed by this estimator, we show in Proposition 4.5.1 that the estimates m and $\tilde{\sigma}$ obtained with the sequential Bernoulli estimator satisfy the expectation bounds $\mathbb{E}[1/m] \leq 1/\mu$, $\mathbb{E}[\tilde{\sigma}] \leq \sigma$ and $\mathbb{E}[1/\sqrt{m}] \leq 1/\sqrt{\mu}$.

(ϵ, δ) -Approximation with information on the coefficient of variation. We explain how the knowledge of a non-increasing function f such that $f(\mu) \geq \sigma/\mu$ can help to solve the (ϵ, δ) -approximation problem (Theorem 4.5.5). If $f(\mu)$ were known, it would suffice to run the quantum sub-Gaussian estimator with $t = f(\mu)/\epsilon \cdot \log(1/\delta)$. We cannot find $f(\mu)$ exactly, but we show how to compute a value \mathcal{V} such that $f(\mu) \leq \mathcal{V} \leq f(c\mu)$ for some constant c with high probability. Our approach is again based on sequential analysis. We instantiate the quantum sub-Gaussian estimator with $t = f(2^{-\ell}) \log(2^\ell/\delta)$ to obtain an estimate $\tilde{\mu}_\ell$ of μ for increasing values of ℓ . We stop this process when we observe a value $\tilde{\mu}_\ell$ that is larger than $2^{-\ell}$, and we take $\mathcal{V} \approx f(2^{-\ell})$. It is easy to see that when ℓ is sufficiently large the estimate $\tilde{\mu}_\ell$ is close to μ by property of the sub-Gaussian estimator. More precisely, if $2^{-\ell} \leq \mu/2$ then we have $\tilde{\mu}_\ell \geq (3/4)\mu \geq 2^{-\ell}$ with high probability. Hence the algorithm is likely to stop with $\ell \leq \log(2/\mu)$. The challenging part is to prove that it does not stop too early. For this purpose, we use another property of the quantum sub-Gaussian estimator, which says that the output estimate is smaller than $(2 + 2\pi)^2\mu$ with high probability *whatever* t is (Theorem 4.4.2). This can be seen as a weak version of Markov's inequality. It implies that when $(2 + 2\pi)^2\mu \leq 2^{-\ell}$ the estimate $\tilde{\mu}_\ell$ is smaller than $2^{-\ell}$ with high probability. Hence the algorithm is likely to stop with $\ell \geq \log(1/((2 + 2\pi)^2\mu))$.

Lower bounds. We sketch the proof of optimality of the quantum sub-Gaussian estimator (Theorem 4.6.2). The lower bound is proved in the stronger quantum query model, which allows us to extend it to all the other models mentioned in Section 4.2. Our approach is inspired by the truncation level chosen in the algorithm. Given σ and t , we consider the two distributions p_0 and p_1 that output respectively $\frac{t\sigma}{\sqrt{1-1/t^2}}$ and $\frac{-t\sigma}{\sqrt{1-1/t^2}}$ with probability $1/t^2$, and 0 otherwise. The two distributions have variance σ^2 and the distance between their means is larger than $\frac{2\sigma}{t}$. Thus, any estimator that satisfies the bound $\Pr[|\tilde{\mu} - \mu| > \frac{\sigma}{t}] \leq \frac{1}{3}$ can distinguish between p_0 and p_1 with constant success probability. However, we show by a reduction to Quantum Search that it requires at least $\Omega(t)$ quantum experiments to distinguish between two distributions that differ with probability at most $1/t^2$.

4.2 Model of input

The input to the mean estimation problem is represented by a real-valued random variable X defined on some probability space. A classical estimator accesses this input by obtaining t *i.i.d* samples of X . In this section, we describe the access model for quantum estimators and we compare it to previous models suggested in the literature. We only consider finite probability spaces for finite encoding reasons. First, we recall the definition of a random variable, and we define a classical model of access called a *random experiment*.

Definition 4.2.1 (RANDOM VARIABLE). A finite *random variable* is a function $X : \Omega \rightarrow E$ for some probability space (Ω, p) , where Ω is a finite sample set, $p : \Omega \rightarrow [0, 1]$ is a probability mass function and $E \subset \mathbb{R}$ is the support of X . As is customary, we will often omit to mention (Ω, p) when referring to the random variable X .

Definition 4.2.2 (RANDOM EXPERIMENT). Given a random variable X on a probability space (Ω, p) , we define a *random experiment* as the process of drawing a sample $\omega \in \Omega$ according to p and observing the value of $X(\omega)$.

We now introduce the concept of “q-random variable” to represent a quantum process that outputs a real number.

Definition 4.2.3 (Q-RANDOM VARIABLE). A *q-variable* is a triple (\mathcal{H}, U, M) where \mathcal{H} is a finite-dimensional Hilbert space, U is a unitary transformation on \mathcal{H} , and $M = \{M_x\}_{x \in E}$ is a projective measurement on \mathcal{H} indexed by a finite set $E \subset \mathbb{R}$. Given a random variable X on a probability space (Ω, p) , we say that a q-variable (\mathcal{H}, U, M) *generates* X when,

- (1) \mathcal{H} is a finite-dimensional Hilbert space with some basis $\{|\omega\rangle\}_{\omega \in \Omega}$ indexed by Ω .
- (2) U is a unitary transformation on \mathcal{H} such that $U|0\rangle = \sum_{\omega \in \Omega} \sqrt{p(\omega)}|\omega\rangle$.
- (3) $M = \{M_x\}_x$ is the projective measurement on \mathcal{H} defined by $M_x = \sum_{\omega: X(\omega)=x} |\omega\rangle\langle\omega|$.

A random variable X is a *q-random variable* if it is generated by some q-variable (\mathcal{H}, U, M) .

We stress that the sample space Ω may not be known explicitly, and we do not assume that it is easy to perform a measurement in the $\{|\omega\rangle\}_{\omega \in \Omega}$ basis for instance. Often, we are given a unitary U such that $U|0\rangle = \sum_{x \in E} \sqrt{p(x)}|\psi_x\rangle|x\rangle$ for some unknown garbage unit state $|\psi_x\rangle$, together with the measurement $M = \{I \otimes |x\rangle\langle x|\}_{x \in E}$. In this case, we can consider the q-random variable X defined on the probability space (Ω, p) where $\Omega = \{|\psi_x\rangle|x\rangle\}_{x \in E}$ and $X(|\psi_x\rangle|x\rangle) = x$. This model is similar to previous work [Mon15; HM19; Bel19; GL20]. The much stronger assumption of having a unitary U such that $U|0\rangle = \sum_{x \in E} \sqrt{\Pr[X=x]}|x\rangle$ is studied in [AT07; Bel19] for example.

We make two minimal assumptions on the type of computations that can be performed on the Hilbert space \mathcal{H} . We assume the existence of a comparison oracle (Assumption 4.A) and a rotation oracle (Assumption 4.B). These two oracles have also been used in previous work on quantum mean estimation [Ter99; NW99; BDGT11; Mon15; HM19].

Assumption 4.A (COMPARISON ORACLE). Given a q-random variable X on a probability space (Ω, p) , and any two values $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$ such that $a < b$, there is a unitary operator $C_{a,b}$ acting on $\mathcal{H} \otimes \mathbb{C}^2$ such that for all $\omega \in \Omega$,

$$C_{a,b}(|\omega\rangle|0\rangle) = \begin{cases} |\omega\rangle|1\rangle & \text{when } a < X(\omega) \leq b, \\ |\omega\rangle|0\rangle & \text{otherwise.} \end{cases}$$

Assumption 4.B (ROTATION ORACLE). Given a q-random variable X on a probability space (Ω, p) , and any two values $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$ such that $a < b$, there is a unitary operator $R_{a,b}$ acting on $\mathcal{H} \otimes \mathbb{C}^2$ such that for all $\omega \in \Omega$,

$$R_{a,b}(|\omega\rangle|0\rangle) = \begin{cases} |\omega\rangle \left(\sqrt{1 - \left| \frac{X(\omega)}{b} \right|} |0\rangle + \sqrt{\left| \frac{X(\omega)}{b} \right|} |1\rangle \right) & \text{when } a < X(\omega) \leq b, \\ |\omega\rangle|0\rangle & \text{otherwise.} \end{cases}$$

We now define the measure of complexity used to count the number of accesses to a q-random variable, which are referred to as *quantum experiments*.

Definition 4.2.4 (QUANTUM EXPERIMENT). Let X be a q-random variable that satisfies Assumptions 4.A and 4.B. Let (\mathcal{H}, U, M) be a q-variable that generates X . We define a *quantum experiment* as the process of applying any of the unitaries U , $C_{a,b}$, $R_{a,b}$ (for any values of $a < b$), their inverses or their controlled versions, or performing a measurement according to M .

Note that a random experiment (Definition 4.2.2) can be simulated with two quantum experiments by computing the state $U|0\rangle$ and measuring it according to M . We briefly mention two other possible input models. First, some authors [Gro98; NW99; Hei02; BHH11; CFMW10; BDGT11; LW19] considered the case where p is the uniform distribution and a quantum oracle is provided for the function $\omega \mapsto X(\omega)$. A second model studies the problem of *learning from quantum states* [BJ99; AW18; ABC+20], where the input consists of several copies of $\sum_{x \in E} \sqrt{\Pr[X = x]} |x\rangle$. In this model, it is not possible to use quantum subroutines such as the amplitude estimation algorithm since we do not have access to a unitary preparing the state. In fact, we show in Theorem 4.6.6 that no quantum speedup is achievable for our problem in the latter setting.

4.3 Quantile estimation

In this section, we present a quantum algorithm for estimating the quantiles of a finite random variable X . This is a key ingredient for the sub-Gaussian estimator of Section 4.4. For the convenience of reading, we define a quantile in the following non-standard way (the cumulative distribution function is replaced with its complement).

Definition 4.3.1 (QUANTILE). Given a discrete random variable X and a real $p \in [0, 1]$, the *quantile* of order p is the number $Q(p) = \sup\{x \in \mathbb{R} : \Pr[X \geq x] \geq p\}$.

Our result is inspired by the minimum finding algorithm of Dürr and Høyer [DH96] and its generalization in [AGGW20b]. The problem of estimating the quantiles of a set of numbers under the *uniform* distribution was studied before by Nayak and Wu [NW99; Nay99]. We differ from that work by allowing arbitrary distributions, and by not using the amplitude estimation algorithm. On the other hand, we restrict ourselves to finding a constant factor estimate, whereas [NW99; Nay99] can achieve any wanted accuracy.

The idea behind our algorithm is rather simple: if we compute a sequence of values $-\infty = y_0 \leq y_1 \leq y_2 \leq y_3 \leq \dots$ where each y_{j+1} is sampled from the distribution of X conditioned on $y_{j+1} \geq y_j$, then when $j \simeq \log(1/p)$ the value of y_j should be close to the quantile $Q(p)$. The complexity of sampling each y_j is on the order of $1/\Pr[X \geq y_j]$ classically, but it can be done quadratically faster in the quantum setting. We analyze a slightly different algorithm, where the sequence of samples is strictly increasing and instead of stopping after roughly $\log(1/p)$ iterations we count the number of experiments

performed by the algorithm and stop when it reaches a value close to $1/\sqrt{p}$. This requires showing that the times T_j spent on sampling y_j is neither too large nor too small with high probability, which is proved in the next lemma.

Lemma 4.3.2. *There is a quantum algorithm such that, given a q -random variable X and a value $x \in \mathbb{R} \cup \{-\infty, +\infty\}$, it outputs a sample y from the probability distribution of X conditioned on $y > x$. If we let T denote the number of quantum experiments performed by this algorithm, then there exist two universal constants $c_0 < c_1$ such that $\mathbb{E}[T] \leq c_1/\sqrt{\Pr[X > x]}$ and $\Pr[T < c_0/\sqrt{\Pr[X > x]}] \leq 1/10$.*

Proof. Let (\mathcal{H}, U, M) be a q -variable generating X . We use the comparison oracle $C_{x,+\infty}$ from Assumption 4.A to construct the unitary $V = C_{x,+\infty}(U \otimes I)$ acting on $\mathcal{H} \otimes \mathbb{C}^2$. By definition of $C_{x,+\infty}$ and U (Section 4.2), we have that $V|\mathbf{0}\rangle = \sum_{\omega \in \Omega: X(\omega) \leq x} \sqrt{p(\omega)}|\omega\rangle|0\rangle + \sum_{\omega \in \Omega: X(\omega) > x} \sqrt{p(\omega)}|\omega\rangle|1\rangle = \sqrt{1 - \Pr[X > x]}|\phi_0\rangle|0\rangle + \sqrt{\Pr[X > x]}|\phi_1\rangle|1\rangle$ for some unit states $|\phi_0\rangle, |\phi_1\rangle$ where $|\phi_1\rangle = \frac{1}{\sqrt{\Pr[X > x]}} \sum_{\omega: X(\omega) > x} \sqrt{p(\omega)}|\omega\rangle$. The algorithm for sampling y conditioned on $y > x$ consists of two steps. First, we use the sequential amplitude amplification algorithm $\text{Seq-AAmp}(V, I \otimes |1\rangle\langle 1|)$ from Theorem 3.2.3 on V to obtain the state $|\phi_1\rangle$. Next, we measure $|\phi_1\rangle$ according to M . The claimed properties follow directly from Theorem 3.2.3. \square

We use the next formula for the probability that a value x occurs in the sequence $(y_j)_j$ defined before. This lemma is adapted from [DH96, Lemma 1].

Lemma 4.3.3 (Lemma 47 in [AGGW20b]). *Let X be a discrete random variable. Consider the increasing sequence of random variables Y_0, Y_1, Y_2, \dots where Y_0 is a fixed value and Y_{j+1} for $j \geq 0$ is a sample drawn from X conditioned on $Y_{j+1} > Y_j$. Then, for any $x, y \in \mathbb{R}$,*

$$\Pr[x \in \{Y_1, Y_2, \dots\} \mid Y_0 = y] = \begin{cases} \Pr[X = x \mid X \geq x] & \text{when } x > y, \\ 0 & \text{otherwise.} \end{cases}$$

The quantile estimation algorithm is described in Algorithm 4.1 and its analysis is provided in the next theorem.

1. Repeat the following steps for $i = 1, 2, \dots, \lceil 6 \log(1/\delta) \rceil$.
 - a) Set $y_0 = -\infty$ and initialize a counter $C = 0$ that is incremented each time a quantum experiment is performed.
 - b) Set $j = 1$. Repeat the following process and interrupt it when $C = c'/\sqrt{p}$ (where c' is a constant chosen in the proof of Theorem 4.3.4): sample an element y_{j+1} from X conditioned on $y_{j+1} > y_j$ by using the algorithm of Lemma 4.3.2, set $j \leftarrow j + 1$.
 - c) Set $\tilde{Q}^{(i)} = y_j$.
2. Output $\tilde{Q} = \text{median}(\tilde{Q}^{(1)}, \dots, \tilde{Q}^{(\lceil 6 \log(1/\delta) \rceil)})$.

Algorithm 4.1: Quantile estimation algorithm, $\text{Quantile}(X, p, \delta)$.

Theorem 4.3.4 (QUANTILE ESTIMATION). *Let X be a q -random variable. Given two reals $p, \delta \in (0, 1)$, the approximate quantile \tilde{Q} produced by the quantile estimation algorithm $\text{Quantile}(X, p, \delta)$ (Algorithm 4.1) satisfies*

$$Q(p) \leq \tilde{Q} \leq Q(cp)$$

with probability at least $1 - \delta$, where $c < 1$ is a universal constant. The algorithm performs $O\left(\frac{\log(1/\delta)}{\sqrt{p}}\right)$ quantum experiments.

Proof. Let c_0, c_1 be the universal constants mentioned in Lemma 4.3.2, and set $c = c_0^2/(c_1^2\sqrt{191})$ and $c' = 190c_1$. Fix i and consider the sequence $(y_j)_{j \geq 0}$ that would be computed during the i -th execution of steps 1.a-1.c if the stopping condition on C was removed. We prove that immediately after the c'/\sqrt{p} -th quantum experiment is performed (which may occur during the computation of y_{j+1}), the current value of y_j satisfies $Q(p) \leq y_j \leq Q(cp)$ with probability at least $(9/10)^2$. The analysis is done in two parts.

First, let $x^- = Q(p)$ and denote by T^- the number of experiments performed until y_j becomes larger than or equal to x^- . According to Lemma 4.3.3, the probability that a given x occurs in the sequence $(y_j)_{j \geq 0}$ is equal to $\Pr[X = x \mid X \geq x]$. Moreover, using Lemma 4.3.2, the expected number of experiments performed at step 1.b when $y_j = x$ is at most $c_1/\sqrt{\Pr[X > x]}$. Consequently, we have

$$\mathbb{E}[T^-] \leq c_1 \sum_{x < x^-} \frac{\Pr[X = x \mid X \geq x]}{\sqrt{\Pr[X > x]}}.$$

Suppose that $Q(1) \neq x^-$ (otherwise $T^- = 0$). We upper bound the above sum by splitting it into several parts as follows. Define $Q_k = Q(2^{-k})$ for $k \geq 0$ and let ℓ be the largest integer such that $Q(2^{-\ell}) < x^-$. For each $1 \leq k \leq \ell$ such that $Q_{k-1} \neq Q_k$, we have

$$\begin{aligned} \sum_{Q_{k-1} \leq x < Q_k} \frac{\Pr[X = x \mid X \geq x]}{\sqrt{\Pr[X > x]}} &\leq \frac{1}{\sqrt{\Pr[X > Q_{k-1}]}} + \sum_{Q_{k-1} < x < Q_k} \frac{\Pr[X = x]}{\Pr[X > x]^{3/2}} \\ &\leq \frac{1}{\sqrt{\Pr[X \geq Q_k]}} + \frac{\Pr[X > Q_{k-1}]}{\Pr[X \geq Q_k]^{3/2}} \\ &\leq \frac{1}{\sqrt{2^{-k}}} + \frac{2^{-(k-1)}}{2^{-3k/2}} \\ &\leq 2^{k/2+2}. \end{aligned}$$

Similarly, $\sum_{Q_\ell \leq x < x^-} \frac{\Pr[X = x \mid X \geq x]}{\sqrt{\Pr[X > x]}} \leq 2^{\ell/2} + 2^{-\ell+1}/p^{3/2}$. Thus, $\mathbb{E}[T^-] \leq c_1(\sum_{k=1}^{\ell} 2^{k/2+2} + 2^{\ell/2} + 2^{-\ell+1}/p^{3/2}) \leq 19c_1/\sqrt{p}$ where we used that $\log(1/p) - 1 \leq \ell < \log(1/p)$ since $Q_\ell < Q(p) \leq Q_{\ell+1}$. By Markov's inequality, $\Pr[T^- \leq 190c_1/\sqrt{p}] \geq 9/10$.

Secondly, let $x^+ = Q(cp)$ and denote by T^+ the number of experiments performed at step 1.b to sample y_{j+1} when $y_j \geq x^+$. According to Lemma 4.3.2, we have $\Pr[T^+ \geq c_0/\sqrt{\Pr[X > y_j]}] \geq 9/10$. Moreover, $\Pr[X > y_j] \leq cp = c_0^2/(c_1^2\sqrt{191})p$ by definition of x^+ . Thus, $\Pr[T^+ \geq 191c_1/\sqrt{p}] \geq 9/10$.

We conclude that step 1.b is interrupted when the value $\tilde{Q}^{(i)}$ satisfies $Q(p) \leq \tilde{Q}^{(i)} \leq Q(cp)$ with probability at least $(9/10)^2$. Thus, by the median trick, the output \tilde{Q} satisfies $Q(p) \leq \tilde{Q} \leq Q(cp)$ with probability at least $1 - \delta$. The total number of experiments is guaranteed to be $O(\log(1/\delta)/\sqrt{p})$ by our use of the counter C . \square

4.4 Sub-Gaussian estimator

In this section, we present the main quantum algorithm for estimating the mean of a random variable with a near-quadratic speedup over the classical sub-Gaussian estimators. Our result uses the following *Bernoulli estimator*, which is an easy adaptation of the amplitude estimation algorithm to the mean estimation problem [BHMT02; Ter99; Mon15]. The Bernoulli estimator allows us to estimate the mean of the truncated random variable $X \mathbb{1}_{a < X \leq b}$ for any values of a, b .

Proposition 4.4.1 (BERNOULLI ESTIMATOR). *There exists a quantum algorithm, called the Bernoulli estimator, with the following properties. Let X be a q -random variable and set as input a time parameter $t \geq 0$, two range values $0 \leq a < b$, and a real $\delta \in (0, 1)$ such that $t \geq \log(1/\delta)$. Then, the Bernoulli estimator $\text{BernEst}(X, t, a, b, \delta)$ outputs a mean estimate $\tilde{\mu}_{a,b}$ of $\mu_{a,b} = \mathbb{E}[X \mathbb{1}_{a < X \leq b}]$ such that $|\tilde{\mu}_{a,b} - \mu_{a,b}| \leq \frac{\sqrt{b\mu_{a,b}} \log(1/\delta)}{t} + \frac{b \log(1/\delta)^2}{t^2}$ and $\tilde{\mu}_{a,b} \leq (1 + 2\pi)^2 \mu_{a,b}$ with probability at least $1 - \delta$. It performs $O(t)$ quantum experiments.*

Proof. Let (\mathcal{H}, U, M) be a q -variable generating X . Using the rotation oracle $R_{a,b}$ from Assumption 4.B, we consider the unitary algorithm $V = R_{a,b}(U \otimes I)$ that acts on $\mathcal{H} \otimes \mathbb{C}^2$. In order to simplify notations, let us assume that the random variable X is distributed in the interval (a, b) . Then, $\mu = \mu_{a,b}$ and by definition of $R_{a,b}$ and U (Section 4.2) we have,

$$\begin{aligned} V|0\rangle &= \sum_{\omega \in \Omega} \sqrt{p(\omega)} |\omega\rangle \left(\sqrt{1 - \frac{X(\omega)}{b}} |0\rangle + \sqrt{\frac{X(\omega)}{b}} |1\rangle \right) \\ &= \sqrt{1 - \frac{\mu}{b}} \left(\sum_{\omega \in \Omega} \sqrt{\frac{p(\omega)(b - X(\omega))}{b - \mu}} |\omega\rangle \right) |0\rangle + \sqrt{\frac{\mu}{b}} \left(\sum_{\omega \in \Omega} \sqrt{\frac{p(\omega)X(\omega)}{\mu}} |\omega\rangle \right) |1\rangle. \end{aligned}$$

Thus, there exist some unit states $|\psi_0\rangle, |\psi_1\rangle$ such that $V|0\rangle = \sqrt{1 - \frac{\mu}{b}} |\psi_0\rangle + \sqrt{\frac{\mu}{b}} |\psi_1\rangle$ and $(I \otimes |1\rangle\langle 1|)V|0\rangle = \sqrt{\frac{\mu}{b}} |\psi_1\rangle$. If X takes values outside the interval (a, b) then the same result holds with $\mu_{a,b}$ in place of μ and a different definition of $|\psi_0\rangle, |\psi_1\rangle$.

Consider the output \tilde{v} of the amplitude estimation algorithm $\text{AEst}(V, \Pi, \lceil \frac{2\pi t}{\log(1/\delta)} \rceil)$ (Theorem 3.3.1) where $\Pi = I \otimes |1\rangle\langle 1|$. Then, the estimate $b\tilde{v}$ satisfies the statement of the proposition with probability $8/\pi^2$ by Theorem 3.3.1 and Corollary 3.3.2. The Bernoulli estimator consists of running $\lceil 6 \log(1/\delta) \rceil$ copies of $\text{AEst}(V, \Pi, \lceil \frac{2\pi t}{\log(1/\delta)} \rceil)$ and outputting the median of the results. The success probability is at least $1 - \delta$ by the median trick. \square

The Bernoulli estimator can estimate the mean of a non-negative q -random variable X by setting $a = 0$ and $b = \max X$. However, its performance is worse than that of the classical sub-Gaussian estimators when the maximum of X is large compared to its variance. Our quantum sub-Gaussian estimator (Algorithm 4.2) uses the Bernoulli estimator in a more subtle way, and in combination with the quantile estimation algorithm.

Theorem 4.4.2 (SUB-GAUSSIAN ESTIMATOR). *Let X be a q -random variable with mean μ and variance σ^2 . Given a time parameter t and a real $\delta \in (0, 1)$ such that $t \geq \log(1/\delta)$, the sub-Gaussian estimator $\text{SubGaussEst}(X, t, \delta)$ (Algorithm 4.2) outputs a mean estimate $\tilde{\mu}$ such that,*

$$\Pr \left[|\tilde{\mu} - \mu| \leq \frac{\sigma \log(1/\delta)}{t} \right] \geq 1 - \delta.$$

The algorithm performs $O(t \log^{3/2}(t) \log \log(t))$ quantum experiments. Moreover, if X is non-negative then $\Pr[\tilde{\mu} \leq (2 + 2\pi)^2 \mu] \geq 1 - \delta$.

1. Set $k = \log t$ and $s = dt\sqrt{\log t} \frac{\log(9k/\delta)}{\log(1/\delta)}$, where $d > 1$ is a constant chosen in the proof of Theorem 4.4.2 (if k is not an integer, round t to the next power of two).
2. Compute the median η of $\lceil 30 \log(2/\delta) \rceil$ classical samples from X and define the non-negative random variables

$$Y^+ = (X - \eta)\mathbb{1}_{X \geq \eta} \quad \text{and} \quad Y^- = -(X - \eta)\mathbb{1}_{X \leq \eta}.$$

3. Compute an estimate $\tilde{\mu}_{Y^+}$ of $\mathbb{E}[Y^+]$ and an estimate $\tilde{\mu}_{Y^-}$ of $\mathbb{E}[Y^-]$ by executing the following steps with $Y := Y^+$ and $Y := Y^-$ respectively:
 - a) Compute an estimate \tilde{Q} of the quantile of order $p = \left(\frac{\log(1/\delta)}{6t}\right)^2$ of Y with failure probability $\delta/8$ by using the **quantile estimation** algorithm $\text{Quantile}(Y, p, \delta/8)$.
 - b) Define $a_{-1} = 0$ and $a_\ell = \frac{2^\ell}{t}\tilde{Q}$ for $\ell \geq 0$. Compute an estimate $\tilde{\mu}_\ell$ of $\mathbb{E}[Y\mathbb{1}_{a_{\ell-1} < Y \leq a_\ell}]$ with failure probability $\delta/(9k)$ for each $0 \leq \ell \leq k$, by using the **Bernoulli estimator** $\text{BernEst}(Y, s, a_{\ell-1}, a_\ell, \delta/(9k))$ with s quantum experiments.
 - c) Set $\tilde{\mu}_Y = \sum_{\ell=0}^k \tilde{\mu}_\ell$.
4. Output $\tilde{\mu} = \eta + \tilde{\mu}_{Y^+} - \tilde{\mu}_{Y^-}$.

Algorithm 4.2: Sub-Gaussian estimator, $\text{SubGaussEst}(X, t, \delta)$.

Proof. First, by standard concentration inequalities, the median η computed at step 2 satisfies $|\eta - \mu| \leq 2\sigma$ with probability at least $1 - \delta/2$. Moreover, if $|\eta - \mu| \leq 2\sigma$ then $\sqrt{\mathbb{E}[(X - \eta)^2]} = \sqrt{\mathbb{E}[(X - \mu + \mu - \eta)^2]} \leq \sqrt{\mathbb{E}[(X - \mu)^2]} + |\mu - \eta| \leq 3\sigma$, by using the triangle inequality. Below we prove that for any non-negative random variable Y the estimate $\tilde{\mu}_Y$ of $\mu_Y = \mathbb{E}[Y]$ computed at step 3 satisfies

$$|\tilde{\mu}_Y - \mu_Y| \leq \frac{\sqrt{\mathbb{E}[Y^2]} \log(1/\delta)}{5t} \quad (4.4)$$

with probability at least $1 - \delta/4$. Using the fact that $X = \eta + Y^+ - Y^-$ and $(X - \eta)^2 = Y_+^2 + Y_-^2$, we can conclude that

$$|\tilde{\mu} - \mu| \leq \frac{\left(\sqrt{\mathbb{E}[Y_+^2]} + \sqrt{\mathbb{E}[Y_-^2]}\right) \log(1/\delta)}{5t} \leq \frac{\sqrt{2\mathbb{E}[(X - \eta)^2]} \log(1/\delta)}{5t} \leq \frac{\sigma \log(1/\delta)}{t}$$

with probability at least $1 - \delta$. The algorithm performs $O(\log(1/\delta)) \leq O(t)$ classical experiments during step 2, $O(\log(1/\delta)/\sqrt{p}) \leq O(t)$ quantum experiments during step 3.a, and $O(ks) \leq O(t \log^{3/2}(t) \log \log(t))$ quantum experiments during step 3.b.

We now turn to the proof of Equation (4.4). We make the assumption that all the subroutines used in step 3 are successful, which is the case with probability at least $(1 - \delta/8)(1 - \delta/(9k))^{k+1} \geq 1 - \delta/4$. First, according to Theorem 4.3.4, we have $Q(p) \leq \tilde{Q} \leq Q(cp)$ for some universal constant c . It implies that $cp \leq \Pr[Y \geq Q(cp)] \leq \Pr[Y \geq \tilde{Q}] \leq \mathbb{E}[Y^2]/\tilde{Q}^2$, where the first two inequalities are by definition of the quantile function Q ,

and the last inequality is a standard fact. Consequently, by our choice of p ,

$$\tilde{Q} \leq \frac{6t\sqrt{\mathbb{E}[Y^2]}}{\sqrt{c}\log(1/\delta)}. \quad (4.5)$$

Next, we upper bound the expectation of the part of Y that is above the largest threshold $a_k = \tilde{Q}$ considered in step 3.b. By Cauchy–Schwarz’ inequality, we have $\mathbb{E}[Y \mathbb{1}_{Y > \tilde{Q}}] \leq \sqrt{\mathbb{E}[Y^2] \Pr[Y > \tilde{Q}]}$. Moreover, by definition of Q , $\Pr[Y > \tilde{Q}] \leq \Pr[Y > Q(p)] \leq p$. Thus,

$$\mathbb{E}[Y \mathbb{1}_{Y > \tilde{Q}}] \leq \frac{\sqrt{\mathbb{E}[Y^2] \log(1/\delta)}}{6t}. \quad (4.6)$$

The expectation of Y is decomposed into the sum $\mu_Y = \sum_{\ell=0}^k \mu_\ell + \mathbb{E}[Y \mathbb{1}_{Y > a_k}]$, where $\mu_\ell = \mathbb{E}[Y \mathbb{1}_{a_{\ell-1} < Y \leq a_\ell}]$ is estimated at step 3.b. We have $|\tilde{\mu}_\ell - \mu_\ell| \leq \frac{\sqrt{a_\ell \mu_\ell} \log(1/\delta)}{dt\sqrt{\log t}} + \frac{a_\ell \log(1/\delta)^2}{d^2 t^2 \log t}$ for all $0 \leq \ell \leq k$ according to Proposition 4.4.1. Thus, by the triangle inequality,

$$\begin{aligned} |\tilde{\mu}_Y - \mu_Y| &\leq \sum_{\ell=0}^k |\tilde{\mu}_\ell - \mu_\ell| + \mathbb{E}[Y \mathbb{1}_{Y > a_k}] \\ &\leq \sum_{\ell=0}^k \frac{\sqrt{a_\ell \mu_\ell} \log(1/\delta)}{dt\sqrt{\log t}} + \sum_{\ell=0}^k \frac{a_\ell \log(1/\delta)^2}{d^2 t^2 \log t} + \mathbb{E}[Y \mathbb{1}_{Y > a_k}] \\ &\leq \frac{\tilde{Q} \log(1/\delta)}{dt^2 \sqrt{\log t}} + \sum_{\ell=1}^k \frac{\sqrt{2\mathbb{E}[Y^2 \mathbb{1}_{a_{\ell-1} < Y \leq a_\ell}] \log(1/\delta)}}{dt\sqrt{\log t}} + \frac{2\tilde{Q} \log(1/\delta)^2}{d^2 t^2 \log t} + \mathbb{E}[Y \mathbb{1}_{Y > a_k}] \\ &\leq \frac{\sqrt{2k} \sqrt{\sum_{\ell=1}^k \mathbb{E}[Y^2 \mathbb{1}_{a_{\ell-1} < Y \leq a_\ell}] \log(1/\delta)}}{dt\sqrt{\log t}} + \frac{3\tilde{Q} \log(1/\delta)^2}{dt^2 \sqrt{\log t}} + \mathbb{E}[Y \mathbb{1}_{Y > a_k}] \\ &\leq \frac{\sqrt{2k} \sqrt{\mathbb{E}[Y^2] \log(1/\delta)}}{dt\sqrt{\log t}} + \frac{3\tilde{Q} \log(1/\delta)^2}{dt^2 \sqrt{\log t}} + \mathbb{E}[Y \mathbb{1}_{Y > a_k}] \\ &\leq \frac{\sqrt{2} \sqrt{\mathbb{E}[Y^2] \log(1/\delta)}}{dt} + \frac{18\sqrt{\mathbb{E}[Y^2] \log(1/\delta)}}{\sqrt{c} dt \sqrt{\log t}} + \frac{\sqrt{\mathbb{E}[Y^2] \log(1/\delta)}}{6t} \\ &\leq \frac{\sqrt{\mathbb{E}[Y^2] \log(1/\delta)}}{5t} \end{aligned}$$

where the third step uses $a_0 \mu_0 \leq a_0^2 = (\tilde{Q}/t)^2$ and $a_\ell \mu_\ell \leq (a_\ell/a_{\ell-1}) \mathbb{E}[Y^2 \mathbb{1}_{a_{\ell-1} < Y \leq a_\ell}] \leq 2\mathbb{E}[Y^2 \mathbb{1}_{a_{\ell-1} < Y \leq a_\ell}]$ when $\ell \geq 1$, the fourth step uses the Cauchy–Schwarz inequality, the sixth step uses Equations (4.5) and (4.6), and in the last step we choose $d = 600/\sqrt{c}$.

Finally, we prove that $\Pr[\tilde{\mu} > (2 + 2\pi)^2 \mu] \leq \delta$ when X is non-negative. For any non-negative random variable Y , the estimate $\tilde{\mu}_Y$ of $\mu_Y = \mathbb{E}[Y]$ computed at step 3 is a linear combination of the estimates $\tilde{\mu}_\ell$ obtained with the Bernoulli estimator, each of which satisfies $\tilde{\mu}_\ell \leq (1 + 2\pi)^2 \mu_\ell$ with probability at least $1 - \delta/(9k)$ by Proposition 4.4.1. It implies that $\Pr[\tilde{\mu}_Y \leq (1 + 2\pi)^2 \mu_Y] \geq 1 - \delta/4$. We also have that $\Pr[\eta \leq (3 + 4\pi)\mu] \geq 1 - \delta/2$ by Markov’s inequality and the median trick. Consequently, $\tilde{\mu} \leq \eta + \tilde{\mu}_{Y^+} \leq (3 + 4\pi)\mu + (1 + 2\pi)^2 \mathbb{E}[X \mathbb{1}_{X \geq \eta}] \leq (2 + 2\pi)^2 \mu$ with probability at least $1 - \delta$. \square

4.5 (ϵ, δ) -Estimators

In this section, we study the (ϵ, δ) -approximation problem under two different scenarios. First, we give in Section 4.5.1 a parameter-free estimator that performs *in expectation*

$\tilde{O}\left(\left(\frac{\sigma}{\epsilon\mu} + \frac{1}{\sqrt{\epsilon\mu}}\right) \log\left(\frac{1}{\delta}\right)\right)$ quantum experiments for any random variable distributed in $[0, 1]$. Then, we describe in Section 4.5.2 an estimator that performs $\tilde{O}\left(\frac{\mathcal{V}}{\epsilon} \log(1/\delta)\right)$ quantum experiments given an upper bound \mathcal{V} on the coefficient of variation $|\sigma/\mu|$. We also explain how to find such a bound \mathcal{V} given a non-increasing function f such that $f(\mu) \geq \sigma/\mu$.

4.5.1 Parameter-free estimators

We follow an approach similar to the classical \mathcal{AA} algorithm described in [DKLR00]. We first give a sequential estimator that performs $O(1/(\epsilon\sqrt{\mu}))$ quantum experiments in expectation. We use the term “sequential” in reference to sequential analysis techniques. The classical counterpart of this estimator is the Stopping Rule Algorithm in [DKLR00].

Proposition 4.5.1 (SEQUENTIAL BERNOULLI ESTIMATOR). *There is a quantum algorithm, called the sequential Bernoulli estimator, with the following properties. Let X be a q -random variable distributed in $[0, 1]$ with mean μ . Fix two reals $\epsilon, \delta \in (0, 1/2)$. Then, the sequential Bernoulli estimator $\text{Seq-BernEst}(X, \epsilon, \delta)$ outputs a mean estimate $\tilde{\mu}$ and performs a number T of quantum experiments such that,*

- (1) $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$.
- (2) $\mathbb{E}[1/\tilde{\mu}] \leq O(1/\mu)$ and $\mathbb{E}[\sqrt{\tilde{\mu}}] \leq O(\sqrt{\mu})$.
- (3) $\mathbb{E}[T] \leq O\left(\frac{\log(1/\delta)}{\epsilon\sqrt{\mu}}\right)$.
- (4) *There is a universal constant c such that $\Pr\left[T > c \frac{\log(1/\delta)}{\epsilon\sqrt{\mu}}\right] \leq \delta$.*

Proof. The algorithm consists of running the sequential amplitude estimation algorithm $\text{Seq-AEst}(V, \Pi, \epsilon, \delta)$ (Theorem 3.3.3), where V and Π are given in the proof of Proposition 4.4.1 for $a = 0$ and $b = 1$. The results follow from Theorem 3.3.3. \square

We now describe an algorithm that improves the dependence on ϵ compared to the above estimator. We later show in Proposition 4.6.4 that it is nearly optimal. The classical counterpart of this estimator is the \mathcal{AA} Algorithm in [DKLR00].

1. Compute an estimate m of $\mu = \mathbb{E}[X]$ with relative error $1/2$ by using the **sequential Bernoulli estimator** $\text{Seq-BernEst}(X, 1/2, \delta/4)$.
2. Let Y denote the random variable $(X - X')^2/2$ where X' is independent of X and identically distributed. Compute an estimate $\tilde{\sigma}^2$ of $\sigma^2 = \mathbb{E}[Y]$ with relative error $1/2$ by using the **sequential Bernoulli estimator** $\text{Seq-BernEst}(Y, 1/2, \delta/4)$. Stop the computation if it performs more than $\frac{4c \log(4/\delta)}{\sqrt{\epsilon m}}$ quantum experiments and set $\tilde{\sigma} = 0$ (where c is the constant mentioned in part (4) of Proposition 4.5.1).
3. Compute a second estimate $\tilde{\mu}$ of μ by using the **sub-Gaussian estimator** $\text{SubGaussEst}(X, t, \delta/4)$ with $t = 2 \max\left(\frac{\tilde{\sigma}}{\epsilon m}, \frac{1}{\sqrt{\epsilon m}}\right) \log(4/\delta)$. Output $\tilde{\mu}$.

Algorithm 4.3: Sequential relative estimator.

Theorem 4.5.2 (SEQUENTIAL RELATIVE ESTIMATOR). *Let X be a q -random variable distributed in $[0, 1]$ with mean μ and variance σ^2 . Given two reals $\epsilon, \delta \in (0, 1)$ the estimate $\tilde{\mu}$ output by the sequential relative estimator (Algorithm 4.3) satisfies $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$. The algorithm performs*

$$\tilde{O}\left(\left(\frac{\sigma}{\epsilon\mu} + \frac{1}{\sqrt{\epsilon\mu}}\right) \log(1/\delta)\right)$$

quantum experiments in expectation.

Proof. We assume that $|m - \mu| \leq \mu/2$ at step 1 of the algorithm, which is the case with probability at least $1 - \delta/4$ by Proposition 4.5.1. The analysis of steps 2 and 3 is split into two cases. First, if $\sigma \leq \sqrt{\epsilon\mu}$, then we directly consider the second term in the max at step 3. By Theorem 4.4.2, the estimate $\tilde{\mu}$ satisfies $|\tilde{\mu} - \mu| \leq \frac{\sigma}{2\sqrt{\epsilon m}} \leq \epsilon\mu$ with probability at least $1 - \delta/4$. Secondly, if $\sigma \geq \sqrt{\epsilon\mu}$, then by Proposition 4.5.1 the estimate $\tilde{\sigma}^2$ computed at step 2 satisfies $|\tilde{\sigma}^2 - \sigma^2| \leq \sigma^2/2$ with probability at least $1 - \delta/4$ if we remove the stopping condition. Since we assumed that $m \leq (3/2)\mu$, the computation is stopped if it performs more than $\frac{4c \log(4/\delta)}{\sqrt{\epsilon m}} \geq \frac{2c \log(4/\delta)}{\sigma}$ experiments. However, by Proposition 4.5.1, the number of experiments performed by the sequential Bernoulli estimator at step 2 is at most $\frac{2c \log(4/\delta)}{\sigma}$ with probability at least $1 - \delta/4$. Consequently, the computation is not stopped and $|\tilde{\sigma}^2 - \sigma^2| \leq \sigma^2/2$ with probability at least $1 - \delta/2$. In this case, by considering the first term in the max at step 3, the estimate $\tilde{\mu}$ satisfies $|\tilde{\mu} - \mu| \leq \frac{\sigma}{2\tilde{\sigma}/(\epsilon m)} \leq \epsilon\mu$ with probability at least $1 - \delta/4$. The overall success probability is at least $1 - \delta$.

We now analyze the expected number of quantum experiments performed by the algorithm. Step 1 performs $O(\log(1/\delta)/\sqrt{\mu})$ experiments in expectation by Proposition 4.5.1. Step 2 is stopped after $O(\log(1/\delta)/(\sqrt{\epsilon\mu}))$ experiments in expectation since $\mathbb{E}[1/\sqrt{m}] \leq O(1/\sqrt{\mu})$ by Proposition 4.5.1. Step 3 performs $\tilde{O}\left(\max\left(\frac{\tilde{\sigma}}{\epsilon m}, \frac{1}{\sqrt{\epsilon m}}\right) \log(1/\delta)\right)$ experiments by Theorem 4.4.2. The estimates $\tilde{\sigma}$ and m are independent if we ignore the stopping condition at step 2, in which case $\mathbb{E}\left[\frac{\tilde{\sigma}}{m}\right] = \mathbb{E}[\tilde{\sigma}] \mathbb{E}\left[\frac{1}{m}\right] \leq O\left(\frac{\sigma}{\mu}\right)$ by Proposition 4.5.1. The stopping condition can only decrease this quantity. Thus, step 3 performs $\tilde{O}\left(\max\left(\frac{\sigma}{\epsilon\mu}, \frac{1}{\sqrt{\epsilon\mu}}\right) \log(1/\delta)\right)$ experiments in expectation. \square

4.5.2 Parametrization by the coefficient of variation

We study the (ϵ, δ) -approximation problem when some information is available on the coefficient of variation $|\sigma/\mu|$. First, if we have an upper bound \mathcal{V} on the coefficient of variation then it suffices to use the sub-Gaussian estimator with the correct parameters.

Corollary 4.5.3 (RELATIVE ESTIMATOR). *There exists a quantum algorithm with the following properties. Let X be a q -random variable with mean μ and variance σ^2 , and set as input a value $\mathcal{V} \geq |\sigma/\mu|$ and two reals $\epsilon, \delta \in (0, 1)$. Then, the algorithm outputs a mean estimate $\tilde{\mu}$ such that $\Pr[|\tilde{\mu} - \mu| > \epsilon|\mu|] \leq \delta$ and it performs*

$$\tilde{O}\left(\frac{\mathcal{V}}{\epsilon} \log(1/\delta)\right)$$

quantum experiments.

Proof. The algorithm runs the **sub-Gaussian estimator** $\text{SubGaussEst}(X, \frac{\mathcal{V}}{\epsilon} \log(1/\delta), \delta)$. \square

This approach has the advantage that the number of experiments does not scale with $1/\sqrt{\mu}$ as in Theorem 4.5.2. On the other hand, it requires us to know a “good” upper

bound on the coefficient of variation. This assumption is often met in practice, as is the case for the applications considered in Chapters 6 and 9.

We now consider the weaker hypothesis where, instead of a direct upper bound \mathcal{V} on σ/μ , we have a non-increasing function f such that $f(\mu) \geq \sigma/\mu$. We first describe an algorithm to decide if the mean μ lies in an interval $[\alpha, \beta]$ or is ϵ -far from it. Our approach crucially relies on the inequality $\tilde{\mu} \leq (2 + 2\pi)^2 \mu$ proved for the quantum sub-Gaussian estimator (Theorem 4.4.2).

1. Compute an estimate $\tilde{\mu}$ of $\mu = \mathbb{E}[X]$ by using the **sub-Gaussian estimator** $\text{SubGaussEst}(X, t, \delta)$ with $t = f\left(\frac{\alpha}{8(1+\pi)^2}\right) \frac{4 \log(1/\delta)}{\epsilon}$.
2. If $\tilde{\mu} \in [(1 - \frac{\epsilon}{2})\alpha, (1 + \frac{\epsilon}{2})\beta]$ then output $B = 1$, else output $B = 0$.

Algorithm 4.4: Interval estimator, $\text{IntervEst}(X, f, \alpha, \beta, \epsilon, \delta)$.

Theorem 4.5.4 (INTERVAL ESTIMATOR). *Let X be a q -random variable distributed in $[0, 1]$ with mean μ and variance σ^2 , and set as input a non-increasing function f such that $f(\mu) \geq \sigma/\mu$, two endpoints $\alpha < \beta$ and two reals $\epsilon, \delta \in (0, 1)$. Then, the output B of the interval estimator $\text{IntervEst}(X, f, \alpha, \beta, \epsilon, \delta)$ (Algorithm 4.4) satisfies,*

- (1) *If $\mu \in [\alpha, \beta]$ then $B = 1$ with probability at least $1 - \delta$.*
- (2) *If $\mu \notin [(1 - \epsilon)\alpha, (1 + \epsilon)\beta]$ then $B = 0$ with probability at least $1 - \delta$.*

The algorithm performs $\tilde{O}\left(f\left(\frac{\alpha}{8(1+\pi)^2}\right) \frac{\log(1/\delta)}{\epsilon}\right)$ quantum experiments.

Proof. Assume first that $\mu \geq \frac{\alpha}{8(1+\pi)^2}$. Then, $t \geq 4\sigma/(\epsilon\mu) \log(1/\delta)$ and, by property of the sub-Gaussian estimator, $|\tilde{\mu} - \mu| \leq (\epsilon/4)\mu$ with probability at least $1 - \delta$. Thus, with probability at least $1 - \delta$, if $\mu \in [\alpha, \beta]$ then $\tilde{\mu} \in [(1 - \frac{\epsilon}{4})\alpha, (1 + \frac{\epsilon}{4})\beta]$, if $\mu \geq (1 + \epsilon)\beta$ then $\tilde{\mu} \geq (1 - \epsilon/4)\mu \geq (1 - \epsilon/4)(1 + \epsilon)\beta \geq (1 + \epsilon/2)\beta$, and if $\mu \leq (1 - \epsilon)\alpha$ then $\tilde{\mu} \leq (1 + \epsilon/4)\mu \leq (1 + \epsilon/4)(1 - \epsilon)\alpha \leq (1 - \epsilon/2)\alpha$. In all three cases, the output B is correct with probability at least $1 - \delta$.

Assume now that $\mu \leq \frac{\alpha}{8(1+\pi)^2}$. By property of the sub-Gaussian estimator (second part of Theorem 4.4.2), we have that $\tilde{\mu} \leq (2 + 2\pi)^2 \mu \leq \frac{(2+2\pi)^2}{8(1+\pi)^2} \alpha = \alpha/2 \leq (1 - \epsilon/2)\alpha$ with probability at least $1 - \delta$, in which case the output is $B = 0$. \square

We use the above algorithm to compute a value \mathcal{V} that approximates $f(\mu)$. This value can be used subsequently in Corollary 4.5.3 to estimate the mean with any desired accuracy.

Theorem 4.5.5. *Let X be a q -random variable distributed in $[0, 1]$ with mean μ and variance σ^2 , and set as input a non-increasing function f such that $f(\mu) \geq \sigma/\mu$ and a real $\delta \in (0, 1)$. Then, for any integer $k \geq 1$ and some universal constant c , Algorithm 4.5 outputs a number \mathcal{V} such that*

$$f(\mu) \leq \mathcal{V} \leq f(c2^{-k}\mu)$$

and it performs $\tilde{O}(\mathcal{V} \log(2^k/\mu)^2 \log(1/\delta))$ quantum experiments, with probability $1 - \delta^k$.

1. Set $\ell = 1$.
2. Run the **interval estimator** $\text{IntervEst}(X, f, \alpha, \beta, \epsilon, \delta)$ with $\alpha = 2^{-\ell}$, $\beta = +\infty$, $\epsilon = 1/2$ and $\delta/2^\ell$. Let B denote the output of the algorithm.
3. If $B = 1$ then set $\ell = \ell + 1$ and go to step 2. Else, output $\mathcal{V} = f(2^{-\ell-1})$.

Algorithm 4.5: Computing an upper bound on the coefficient of variation.

Proof. We first assume that all calls to the interval estimator at step 2 of Algorithm 4.5 are successful, which is the case with probability at least $1 - \sum_{\ell=1}^{\infty} \delta/2^\ell \geq 1 - \delta$. According to Theorem 4.5.4, we have $B = 0$ when $\ell < \log(1/(2\mu))$ and $B = 1$ when $\ell \geq \log(1/\mu)$. Consequently, the algorithm stops when ℓ is between $\log(1/(2\mu))$ and $\lceil \log(1/\mu) \rceil + 1$. In this case, the output $\mathcal{V} = f(2^{-\ell-1})$ satisfies $f(\mu) \leq \mathcal{V} \leq f(\mu/8)$ and the algorithm performs at most $\sum_{i=1}^{\lceil \log(1/\mu) \rceil + 1} \tilde{O}\left(f\left(\frac{2^{-i}}{8(1+\pi)^2}\right) \log(2^i/\delta)\right) \leq \tilde{O}\left(f\left(\frac{\mu}{2^{5(1+\pi)^2}}\right) \log(1/\mu)^2 \log(1/\delta)\right)$ quantum experiments. This proves the theorem in the case $k = 1$.

For $k \geq 2$, each time step 2 is executed with $\ell \geq \log(1/\mu)$ the probability that $B = 0$ is at least $1 - \delta$. Thus, the probability that the algorithm has not yet stopped when $\ell = \lceil \log(1/\mu) \rceil + k$ is at most δ^k . The number of experiments performed up to that point is at most $\sum_{i=1}^{\lceil \log(1/\mu) \rceil + k} \tilde{O}\left(f\left(\frac{2^{-i}}{8(1+\pi)^2}\right) \log(2^i/\delta)\right) \leq \tilde{O}\left(f\left(\frac{2^{-k}\mu}{16(1+\pi)^2}\right) \log(2^k/\mu)^2 \log(1/\delta)\right)$. \square

As an application of the above results, we consider the case where f is a power function. We obtain a simple formula for the number of experiments performed in expectation. This result is used in Chapter 6.

Corollary 4.5.6 (EXPONENTIAL ESTIMATOR). *There is a quantum algorithm, called the exponential estimator, with the following properties. Let X be a q -random variable distributed in $[0, 1]$ with mean μ and variance σ^2 , and set as input a function $f : x \mapsto c/x^d$ and two reals $\epsilon, \delta \in (0, 1)$ such that $f(\mu) \geq \sigma/\mu$ and $\delta < 2^{-3d}$, where $c, d \geq 0$ are two constants. Then, the exponential estimator $\text{ExpEst}(X, f, \epsilon, \delta)$ outputs an estimate $\tilde{\mu}$ such that $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$ and it performs a number T of quantum experiments such that*

$$\mathbb{E}[T] \leq \sqrt{\mathbb{E}[T^2]} \leq \tilde{O}(f(\mu) \cdot (1/\epsilon + \log(1/\mu) \log(1/\delta)) \log(1/\delta)).$$

Proof. The algorithm proceeds as follows: first, it computes \mathcal{V} by using the algorithm of Theorem 4.5.5 with input parameters $f, \delta/2$; then it computes $\tilde{\mu}$ by using the algorithm of Corollary 4.5.3 with input parameters $\mathcal{V}, \epsilon, \delta/2$. The statement $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$ is easy to prove. The number T of experiments satisfies

$$\begin{aligned} \mathbb{E}[T^2] &\leq \tilde{O}\left(\sum_{k=0}^{\infty} \delta^k \cdot \left(f(c2^{-k}\mu) \log(2^k/\mu)^2 \log(1/\delta)^2 + \frac{f(c2^{-k}\mu)}{\epsilon} \log(1/\delta)\right)^2\right) \\ &\leq \tilde{O}\left(\sum_{k=0}^{\infty} \delta^k \cdot \left(f(\mu)2^{kd} \log(2^k/\mu)^2 \log(1/\delta)^2 + \frac{f(\mu)}{\epsilon} 2^{kd} \log(1/\delta)\right)^2\right) \\ &\leq \tilde{O}(f(\mu)^2 (\log(1/\mu)^2 \log(1/\delta)^2 + 1/\epsilon^2) \log(1/\delta)^2) \end{aligned}$$

where the second step uses that $f : x \mapsto c/x^d$, and the last step uses that $\delta < 2^{-3d}$. \square

4.6 Lower bounds

We prove several lower bounds for the mean estimation problem under different scenarios. In Section 4.6.1, we study the number of experiments that must be performed to estimate the mean with a sub-Gaussian error rate. In Section 4.6.2, we study the number of experiments needed to solve the (ϵ, δ) -approximation problem. Finally, in Section 4.6.3, we consider the mean estimation problem in the state-based model, where the input consists of several copies of a quantum state encoding a distribution.

4.6.1 Sub-Gaussian estimation

We show that the quantum sub-Gaussian estimator described in Theorem 4.4.2 is optimal up to a polylogarithmic factor. We make use of the following lower bound for Quantum Search in the small-error regime.

Proposition 4.6.1 (Theorem 4 in [BCWZ99]). *Let $N > 0$, $1 \leq K \leq 0.9N$ and $\delta \geq 2^{-N}$. Let $T(N, K, \delta)$ be the minimum number of quantum queries any algorithm must use to decide with failure probability at most δ whether a function $f : [N] \rightarrow \{0, 1\}$ has 0 or K preimages of 1. Then, $T(N, K, \delta) \geq \Omega(\sqrt{N/K} \log(1/\delta))$.*

We construct two particular probability distributions that allow us to reduce the Quantum Search problem to the sub-Gaussian mean estimation problem.

Theorem 4.6.2. *Let $t > 1$ and $\delta \in (0, 1)$ such that $t \geq 2 \log(1/\delta)$. Fix $\sigma > 0$ and consider the family \mathcal{P}_σ of all q -random variables with variance σ^2 . Let $T(t, \sigma, \delta)$ be the minimum number of quantum experiments any algorithm must perform to compute with failure probability at most δ a mean estimate $\tilde{\mu}$ such that $|\tilde{\mu} - \mu| \leq \frac{\sigma \log(1/\delta)}{t}$ for any $X \in \mathcal{P}_\sigma$ with mean μ . Then, $T(t, \sigma, \delta) \geq \Omega(t)$.*

Proof. Let $s = \frac{t}{\log(1/\delta)}$ and $b = \frac{s}{\sqrt{1-1/s^2}}\sigma$. We define the probability distribution p_0 with support $\{0, b\}$ that takes value b with probability $\frac{1}{s^2}$. Similarly, we define the probability distribution p_1 with support $\{0, -b\}$ that takes value $-b$ with probability $\frac{1}{s^2}$. The variance of each distribution is equal to σ^2 . Moreover, the means μ_0 and μ_1 of the two distributions satisfy that,

$$\mu_0 - \mu_1 > 2 \frac{\sigma \log(1/\delta)}{t}. \quad (4.7)$$

Let N, K be two integers such that $N \geq \log(1/\delta)$ and $K/N = 1/s^2$ (assuming s is rational). Let F_0 be the family of all functions $f : [N] \rightarrow \{0, 1\}$ with exactly K preimages of 1. Similarly, let F_1 be the family of all functions $f : [N] \rightarrow \{-1, 0\}$ with exactly K preimages of -1 . By using Proposition 4.6.1, it is easy to see that any algorithm that can distinguish between $f \in F_0$ and $f \in F_1$ with success probability $1 - \delta$ must use at least $\Omega(\sqrt{N/K} \log(1/\delta)) = \Omega(s \log(1/\delta)) = \Omega(t)$ quantum queries to f . We associate with each function $f \in F_0 \cup F_1$ the q -variable $(\mathcal{H}, U, M)_f$ where $\mathcal{H} = \mathbb{C}^{N+2}$, $U|0\rangle = \frac{1}{\sqrt{N}} \sum_{x \in [N]} |x\rangle |f(x)\rangle$, and $M = \{I \otimes |0\rangle\langle 0|, I \otimes |-1\rangle\langle -1|, I \otimes |1\rangle\langle 1|\}$. The random variable X generated by $(\mathcal{H}, U, M)_f$ is distributed according to p_0 if $f \in F_0$, and according to p_1 if $f \in F_1$. Moreover, one quantum experiment with respect to X can be simulated with one quantum query to f . Consequently, any algorithm that can distinguish between a random variable distributed according to p_0 or p_1 with success probability $1 - \delta$ must perform at least $\Omega(t)$ quantum experiments. On the other hand, by Equation (4.7), if an algorithm can estimate the mean with an error rate smaller than $\frac{\sigma \log(1/\delta)}{t}$ then it can distinguish between $f \in F_0$ and $f \in F_1$. Thus, $T(t, \sigma, \delta) \geq \Omega(t)$. \square

4.6.2 (ϵ, δ) -Estimation

We consider the (ϵ, δ) -estimation problem in the parameter-free setting, when the coefficient of variation is unknown. We make use of the next lower bound for Quantum Counting.

Proposition 4.6.3 (Theorem 4.2.6 in [Nay99]). *Let $N > 0$, $1 < K \leq N$ and $\epsilon \in (\frac{1}{4K}, 1)$. Consider the set of all quantum algorithms such that, given a query oracle to any function $f : [N] \rightarrow \{0, 1\}$, they return an estimate \tilde{C} of the number C of preimages of 1 in f such that $|\tilde{C} - C| \leq \epsilon C$ with probability at least $2/3$. Let $T_K(N, \epsilon)$ be the minimum number of quantum queries any such algorithm must use when the oracle has exactly K preimages of 1. Then, $T_K(N, \epsilon) \geq \Omega\left(\frac{\sqrt{K(N-K)}}{\epsilon K+1} + \sqrt{\frac{N}{\epsilon K+1}}\right)$.*

We obtain by a simple reduction to the above problem that the result described in Theorem 4.5.2 is nearly optimal.

Proposition 4.6.4. *Let $\epsilon \in (0, 1)$. Let $\mathcal{P}_{\mathcal{B}}$ denote the family of all q -random variables that follow a Bernoulli distribution. Consider any algorithm that takes as input $X \in \mathcal{P}_{\mathcal{B}}$ and that outputs a mean estimate $\tilde{\mu}$ such that $|\tilde{\mu} - \mathbb{E}[X]| \leq \epsilon \mathbb{E}[X]$ with probability at least $2/3$. Then, for any $\mu \in (0, 1)$, there exists $X \in \mathcal{P}_{\mathcal{B}}$ with mean μ such that the algorithm performs at least $\Omega\left(\frac{\sigma}{\epsilon\mu} + \frac{1}{\sqrt{\epsilon\mu}}\right)$ quantum experiments on input X , where $\sigma^2 = \text{Var}[X]$.*

Proof. Given $\epsilon \in (0, 1)$ and $\mu \in (0, 1)$, we choose two integers K and N such that $K > 1/(4\epsilon)$ and $K/N = \mu$ (assuming μ is rational). Similarly to the proof of Theorem 4.6.2, we associate with each function $f : [N] \rightarrow \{0, 1\}$ the q -variable $(\mathcal{H}, U, M)_f$ where $\mathcal{H} = \mathbb{C}^{N+2}$, $U|0\rangle = \frac{1}{\sqrt{N}} \sum_{x \in [N]} |x\rangle |f(x)\rangle$, and $M = \{I \otimes |0\rangle\langle 0|, I \otimes |1\rangle\langle 1|\}$. If a quantum algorithm can estimate the mean of any Bernoulli random variable with error ϵ and success probability $2/3$, then it can be used to count the number of preimages of 1 in f with the same accuracy. Thus, by Proposition 4.6.3, any such algorithm must perform at least $\Omega\left(\frac{\sqrt{K(N-K)}}{\epsilon K+1} + \sqrt{\frac{N}{\epsilon K+1}}\right) = \Omega\left(\frac{\sqrt{\mu(1-\mu)}}{\epsilon\mu+1/N} + \frac{1}{\sqrt{\epsilon\mu+1/N}}\right) = \Omega\left(\frac{\sigma}{\epsilon\mu} + \frac{1}{\sqrt{\epsilon\mu}}\right)$ quantum experiments on a q -random variable with mean μ and variance $\sigma^2 = \mu(1-\mu)$. \square

4.6.3 State-based estimation

We consider the *state-based* model where the input consists of several copies of a quantum state $|p\rangle = \sum_{x \in E} \sqrt{p(x)} |x\rangle$ encoding a distribution p over E . This model is weaker than the one described before, since it does not provide access to a unitary algorithm preparing $|p\rangle$. We prove that no quantum speedup is achievable in this setting. Our result uses the next lower bound on the number of copies needed to distinguish two states.

Lemma 4.6.5. *Let $\delta \in (0, 1)$ and consider two probability distributions p_0 and p_1 with the same finite support $E \subset \mathbb{R}$. Define the states $|\phi_0\rangle = \sum_{x \in E} \sqrt{p_0(x)} |x\rangle$ and $|\phi_1\rangle = \sum_{x \in E} \sqrt{p_1(x)} |x\rangle$. Then, the smallest integer T such that there is an algorithm that can distinguish $|\phi_0\rangle^{\otimes T}$ from $|\phi_1\rangle^{\otimes T}$ with success probability at least $1 - \delta$ satisfies $T \geq \frac{\ln(1/(4\delta))}{D(p_0||p_1)}$, where $D(p_0||p_1) = \sum_{x \in E} p_0(x) \ln\left(\frac{p_0(x)}{p_1(x)}\right)$ is the KL-divergence from p_0 to p_1 .*

Proof. According to Helstrom's bound [Hel69] the best success probability to distinguish between two states $|\phi\rangle$ and $|\phi'\rangle$ is $\frac{1}{2}(1 + \sqrt{1 - |\langle \phi | \phi' \rangle|^2})$. Thus, the smallest number T needed to distinguish $|\phi_0\rangle^{\otimes T}$ from $|\phi_1\rangle^{\otimes T}$ must satisfy $\frac{1}{2}(1 + \sqrt{1 - |\langle \phi_0 | \phi_1 \rangle|^{2T}}) \geq 1 - \delta$.

It implies that $T \geq \frac{-\ln(1-(1-2\delta)^2)}{-2\ln(|\langle \phi_0 | \phi_1 \rangle|)} \geq \frac{\ln(1/(4\delta))}{-2\ln\left(\sum_{x \in E} p_0(x) \sqrt{\frac{p_1(x)}{p_0(x)}}\right)} \geq \frac{\ln(1/(4\delta))}{\sum_{x \in E} p_0(x) \ln\left(\frac{p_0(x)}{p_1(x)}\right)} = \frac{\ln(1/(4\delta))}{D(p_0||p_1)}$

where the second inequality uses the concavity of the logarithm function. \square

We use the above lemma to show that no quantum mean estimator can perform better than the classical sub-Gaussian estimators in the state-based input model.

Theorem 4.6.6. *Let $t > 1$ and $\delta \in (0, 1)$ such that $t \geq 2 \log(1/\delta)$. Fix $\sigma > 0$ and consider the family \mathcal{P}_σ of all distributions with finite support whose variance lies in the interval $[\sigma^2, 4\sigma^2]$. For any $p \in \mathcal{P}_\sigma$ with support $E \subset \mathbb{R}$, define the state $|p\rangle = \sum_{x \in E} \sqrt{p(x)}|x\rangle$. Let $T(t, \sigma, \delta)$ be the smallest integer such that there exists an algorithm that receives the state $|p\rangle^{\otimes T(t, \sigma, \delta)}$ for any $p \in \mathcal{P}_\sigma$, and that outputs an estimate $\tilde{\mu}$ of the mean μ of p such that $\Pr\left[|\tilde{\mu} - \mu| > \sqrt{\frac{\sigma^2 \log(1/\delta)}{t}}\right] \leq \delta$. Then, $T(t, \sigma, \delta) \geq \Omega(t)$.*

Proof. Let $s = \frac{t}{\log(1/\delta)}$, $b = \frac{s}{\sqrt{s-1}}\sigma$ and $\alpha = 2 \ln\left(1 + \sqrt{1 - \frac{1}{s}}\right)$. We define the two probability distributions p_0 and p_1 with support $E = \{0, b\}$ where $p_0(b) = \frac{e^\alpha}{s}$ and $p_1(b) = \frac{1}{s}$. Let μ_0 and σ_0^2 (resp. μ_1 and σ_1^2) denote the expectation and the variance of p_0 (resp. p_1). Observe that $\sigma_0 \in [\sigma, 2\sigma]$ and $\sigma_1 = \sigma$, thus $p_0, p_1 \in \mathcal{P}_\sigma$. Moreover, $\mu_0 - \mu_1 > 2\sqrt{\frac{\sigma^2 \log(1/\delta)}{t}}$ since $\mu_0 - \mu_1 = \sigma \frac{e^\alpha - 1}{\sqrt{s-1}} \geq \sigma(e^{\alpha/2} + 1) \frac{e^{\alpha/2} - 1}{\sqrt{s-1}} = \sigma(e^{\alpha/2} + 1) \sqrt{\frac{\log(1/\delta)}{t}}$ and $2\sigma \leq \sigma_0 + \sigma_1 = \sigma\left(\sqrt{\frac{e^\alpha s - e^{2\alpha}}{s-1}} + 1\right) < \sigma(e^{\alpha/2} + 1)$. Thus, we can distinguish $|p_0\rangle^{\otimes T(t, \sigma, \delta)}$ from $|p_1\rangle^{\otimes T(t, \sigma, \delta)}$ with failure probability δ by using any optimal algorithm that satisfies the error bound stated in the theorem. Since the KL-divergence from p_0 to p_1 is $D(p_0 \| p_1) \leq p_0(b) \ln\left(\frac{p_0(b)}{p_1(b)}\right) = \frac{\alpha e^\alpha}{s^2} \leq \frac{6}{s}$, we must have $T(t, \sigma, \delta) \geq \Omega\left(\frac{\log(1/\delta)}{D(p_1 \| p_0)}\right) = \Omega(t)$ by Lemma 4.6.5. \square

4.7 Discussion

One interesting open question is to find a quantum mean estimator that achieves the deviation bound $\Pr\left[|\tilde{\mu} - \mu| > \frac{\sigma \log(1/\delta)}{t}\right] \leq \delta$ by performing a number of experiments that is *linear* in t . The current best upper bound (Theorem 4.4.2) is $O(t \log^{3/2}(t) \log \log(t))$, and the lower bound is $\Omega(t)$ (Theorem 4.6.2). A first step toward this goal could be to obtain a better algorithm for the restricted case of Gaussian distributions. An equivalent goal is to find the smallest value L such that the deviation bound $\Pr\left[|\tilde{\mu} - \mu| > L \frac{\sigma \log(1/\delta)}{t}\right] \leq \delta$ can be achieved by a quantum mean estimator that performs *at most* t quantum experiments. Classically, for the sub-Gaussian deviation bound of Equation (4.1), the optimal value is $L = \sqrt{2}(1 + o(1))$ [Cat12; LV20].

There exist many variants of the quantum mean estimation problem that have not been explored in the quantum model yet. Let us mention for instance the multivariate setting [LM19], where the objective is to estimate the mean of a random variable taking values in \mathbb{R}^d . The first polynomial-time classical algorithm for this problem was only found recently by Hopkins [Hop20].

We present two applications of the quantum sub-Gaussian estimator later in this thesis. In Chapter 6, we describe a quantum query algorithm for counting the number of triangles in a graph. This result uses a variant of the quantum sub-Gaussian estimator developed in the next chapter. In Chapter 9, we describe a quantum streaming algorithm for approximating the frequency moments, based on the relative estimator of Corollary 4.5.3. We note that our quantum sub-Gaussian estimator can also be plugged in simulated annealing algorithms [Mon15; HW20; CCH+19; AHN+20] (though in this case Montanaro's estimator [Mon15] is often sufficient since the variance is small), or for simplifying the non-integer Rényi entropy estimation algorithms described in [LW19, Section V].

5

Variable-Time Mean Estimation

This chapter is based on the following papers:

[HM19] Y. Hamoudi and F. Magniez. “Quantum Chebyshev’s Inequality and Applications”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 69:1–69:16.

[HM21a] Y. Hamoudi and F. Magniez. “Quantum Approximate Triangle Counting”. In submission. 2021.

5.1 Introduction

The mean estimation problem presupposes the existence of a random or quantum process whose output value encodes some given random variable X . The computational complexity of solving the mean estimation problem is proportional to the number of times this process is invoked, and to the execution time of the process itself. The first of these two quantities was studied in the previous chapter, where it was called the “number of (random or quantum) experiments”. It reflects the amount of information that must be gathered about X to estimate its mean. The second quantity, which we call the *stopping time* T , measures the computational cost of performing each experiment. The purpose of the present chapter is to understand the underplay between these two quantities.

The stopping time T of a *random* experiment is defined as the total number of operations performed during its execution. The decision to end a random experiment is often based on the observation of a certain random event during the computation. Thus, the stopping time T is itself modeled as a random variable. The expected time complexity of an algorithm is related to the product of the number of random experiments it performs and of the *average* stopping time $T_1 = \mathbb{E}[T]$, by linearity of expectation. This simple relationship is of crucial importance in amortized analysis. Indeed, it offers a more realistic measure of complexity than by considering the *maximum* stopping time $T_{\max} = \max(T)$.

The stopping time of a *quantum* experiment is more subtle to define. The collapsing property of quantum measurements make the design of adaptive stopping rules more difficult to achieve in this setting. Thus, there is often no difference between the average and the total number of operations performed during a quantum experiment. Ambainis [Amb12] addressed a related issue by introducing the concept of *variable-time algorithm*. A variable-time algorithm has the property that there is a non-negligible probability to obtain the same output whether its memory is observed at an intermediate stage of the computation or at the end of it. The stopping time T of such an algorithm is defined as the random variable distributed according to the probability that a *fictitious* intermediate measurement, at a given step of the computation, would return a completed result. Ambainis gave an

operational meaning to this quantity by showing that, for the particular task of searching a marked item in a database, there exists an algorithm whose complexity scales as the product of the usual number of Grover’s iterations and of the ℓ_2 -average stopping time $T_2 = \sqrt{\mathbb{E}[T^2]}$ of the algorithm generating the database [Amb10b]. He later generalized this result to a *variable-time amplitude amplification* (VTAA) algorithm [Amb10c; Amb12], for the task of projecting the output state of a variable-time algorithm in some chosen subspace.

We adopt the same framework as Ambainis to study the mean estimation problem on a random variable X generated by a variable-time algorithm with stopping time T . We focus on the problem of estimating the mean $\mu = \mathbb{E}[X]$ with relative error ϵ . We exhibit an algorithm whose time complexity scales as the product $\mathcal{V}T_2\epsilon^{-2}$, where $\mathcal{V} \geq \sigma/\mu$ is an upper bound on the coefficient of variation of X and $T_2 \geq \sqrt{\mathbb{E}[T^2]}$ is an upper bound on its ℓ_2 -average stopping time. In comparison, a direct application of the estimators constructed in the previous chapter (Section 4.5) would require roughly $\mathcal{V}T_{\max}\epsilon^{-1}$ operations, and a classical estimator would perform $\mathcal{V}^2T_1\epsilon^{-2}$ operations in expectation. Our main ingredients are a new *variable-time amplitude estimation* (VTAE) algorithm, and a new variant of the mean estimator algorithms studied in the previous chapter.

5.1.1 Related work

We refer the reader to Chapter 4 for related work on the mean estimation problem. We focus here on previous work for variable-time quantum algorithms.

Ambainis initiated the study of quantum algorithms with variable stopping times in [Amb10b]. He considered the problem of finding a marked item among n elements, where the i -th element requires time τ_i to be checked. The case of $\tau_1 = \dots = \tau_n = 1$ corresponds to the standard Grover search algorithm, and a naive generalization to arbitrary stopping times would lead to a complexity of $O(\sqrt{n} \max_i \tau_i)$. Instead, Ambainis obtained an optimal complexity that scales as the ℓ_2 -average $O(\sqrt{\tau_1^2 + \dots + \tau_n^2})$. This result was later generalized to a *variable-time amplitude amplification* (VTAA) algorithm in [Amb10c; Amb12]. The VTAA algorithm prepares a state proportional to $\Pi U |\mathbf{0}\rangle$, where Π is a fixed projector and U is a variable-time algorithm with stopping time T , in time $\tilde{O}(T_{\max} + \frac{1}{\sqrt{p}} \sqrt{\mathbb{E}[T^2]})$ where $p = \|\Pi U |\mathbf{0}\rangle\|^2$. In comparison, the standard amplitude amplification algorithm (Theorem 3.2.1) runs in time $O(\frac{1}{\sqrt{p}} T_{\max})$. Chakraborty, Gilyén and Jeffery [CGJ19] built on Ambainis’ work to develop a *variable-time amplitude estimation* (VTAE) algorithm whose complexity scales as $\tilde{O}(\frac{1}{\epsilon}(T_{\max} + \frac{1}{\sqrt{p}} \sqrt{\mathbb{E}[T^2]}))$ to estimate the aforementioned squared amplitude p with relative error ϵ . In our work, we propose a new version of the VTAE algorithm where the dependence on T_{\max} is logarithmic, and with additional properties that are needed to construct our mean estimator algorithms.

The VTAA algorithm has been used for solving systems of linear equations [Amb10c; Amb12; CKS17; CGJ19], and for finding triangles in a graph [Gal14; GN17]. It has been combined with the VTAE algorithm for solving least squares problems and estimating electrical-network quantities [CGJ19], and for best-arm identification in the multi-armed bandit model [WYLC21].

5.1.2 Contributions and organization

We explain in Section 5.2 what a *variable-time algorithm* $U = U_n \dots U_1$ is (Definition 5.2.1), and we define the *stopping time* T of U (Definition 5.2.2). We adapt these definitions to the mean estimation problem and to q-random variables in Definition 5.2.3.

Next, we describe in Section 5.3 our variable-time amplitude estimation (VTAE) algorithm for estimating with relative error ϵ the squared amplitude $p = \|\Pi U|\mathbf{0}\rangle\|^2$, given a projector Π and a variable-time algorithm U with stopping time T . The time complexity of the algorithm is on the order of $\sqrt{\mathbb{E}[T^2]}/(\epsilon^{3/2}\sqrt{p})$ (Theorem 5.3.1), whereas the standard amplitude estimation algorithm (Theorem 3.3.1) has time complexity $T_{\max}/(\epsilon\sqrt{p})$.

We apply the VTAE algorithm to the mean estimation problem in Section 5.4 by describing new mean estimators whose time complexity is proportional to the ℓ_2 -average stopping time $\sqrt{\mathbb{E}[T^2]}$ of the algorithm generating X . In comparison, the time complexity of the estimators developed in Chapter 4 is proportional to the maximum stopping time T_{\max} . We first describe a variable-time Bernoulli estimator in Section 5.4.1 for estimating the mean of a truncated random variable. We use it in Section 5.4.2 to obtain the next estimator that takes as input an upper bound \mathcal{V} on the coefficient of variation of X . This improves upon the time complexity of the estimator given in Corollary 4.5.3.

Theorem 5.4.4 (Restated). *There exists a quantum algorithm with the following properties. Let X be a q -random variable distributed in $[0, 1]$ with mean μ , variance σ^2 and stopping time T . Set as input two time parameters $\mathcal{V}, T_2 \geq 1$, a lower bound $\alpha \geq 0$, and two reals $\epsilon, \delta \in (0, 1)$. Then, the algorithm outputs a mean estimate $\tilde{\mu}$ such that,*

- (1) *If $\mathcal{V} \geq \sigma/\mu$, $T_2 \geq \sqrt{\mathbb{E}[T^2]}$ and $\mu \geq \alpha$ then $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$.*
- (2) $\Pr[\tilde{\mu} \leq 2\mu] \geq 1 - \delta$.

The time complexity of the estimator is $\tilde{O}(\mathcal{V}T_2 \cdot (\log(\frac{1}{\alpha}) + \frac{1}{\epsilon^2}) \cdot \log^4(T_{\max}) \log(\frac{1}{\delta}))$.

Similarly to Corollary 4.5.6, we adapt the previous result to the case where we have a non-increasing function f such that $f(\mu)$ is an upper bound on the coefficient of variation of X (Proposition 5.4.6). We also remove the need for a lower bound α on μ in the above theorem.

5.1.3 Proof overview

Variable-time amplitude estimation. Before explaining how our VTAE algorithm works, we summarize the approach used in the VTAA algorithm developed by Ambainis [Amb10c; Amb12]. A variable-time algorithm (Definition 5.2.1) is a unitary algorithm U that can be decomposed as a product $U = U_n \cdots U_1$ of n consecutive sub-algorithms, and such that the memory of the algorithm contains (in superposition) a Boolean flag indicating whether the computation is finished or not. Each algorithm U_i can only modify the basis states where the flag is set to 0. The purpose of the VTAA algorithm is to prepare the state $\frac{1}{\|\Pi U|\mathbf{0}\rangle\|} \Pi U|\mathbf{0}\rangle$, for some projector Π , by taking advantage of the fact that each U_i can be run on its own. The main idea behind the VTAA algorithm is to intertwine the n computation steps U_1, \dots, U_n with n amplification steps that increase the amplitude of the sub-state with flag 1 lying in the support of Π . This has the effect of improving the running time over the standard amplitude amplification algorithm if a large portion of $\Pi U|\mathbf{0}\rangle$ is prepared at an early stage of U . We follow a similar approach in our VTAE algorithm for estimating the squared amplitude $p = \|\Pi U|\mathbf{0}\rangle\|^2$. We consider a particular sequence of intermediate values p_1, \dots, p_n that are related to the n amplification steps performed in the VTAA algorithm. We show that the product $p_1 \cdots p_i$ of the first i -th terms is equal to the squared amplitude $\|\Pi U_i \cdots U_1|\mathbf{0}\rangle\|^2$ (Lemma 5.3.3). Moreover, if a large portion of $\Pi U|\mathbf{0}\rangle$ is prepared at an early stage of U then it suffices to estimate $\|\Pi U_i \cdots U_1|\mathbf{0}\rangle\|^2$ for a small value of i to obtain a good approximation of p . Finally, we use the standard amplitude estimation algorithm to compute some estimates $\tilde{p}_1, \dots, \tilde{p}_i$ of the i -th first terms, and we combine them into an estimate $\tilde{p} = \tilde{p}_1 \cdots \tilde{p}_i$ of p (Section 5.3.3).

Variable-time Bernoulli estimator. We modify the Bernoulli estimator presented in Proposition 4.4.1 of Chapter 4 so that it uses the VTAE algorithm instead of the amplitude estimation algorithm. The new *variable-time Bernoulli estimator* (Proposition 5.4.2) can estimate with relative error ϵ the mean of a q-random variable X distributed in $[a, b]$ with a time complexity that depends on b , on the expectation of the truncated random variable $X \mathbb{1}_{a < X \leq b}$, and on the ℓ_2 -average stopping time of X . The main step toward constructing this estimator is to define a new variable-time algorithm U (Proposition 5.4.1) that intertwines the original variable-time algorithm generating X with a controlled rotation operator similar to the one used in the Bernoulli estimator. The normalized expectation $\mathbb{E}[X \mathbb{1}_{a < X \leq b}]/b$ is encoded as a squared amplitude $\|\Pi U |\mathbf{0}\rangle\|^2$ for some projector Π . We then use the VTAE algorithm to estimate the latter quantity with relative error ϵ .

Variable-time (ϵ, δ) -estimator. Our approach (Theorem 5.4.4) is tailored to the (ϵ, δ) -approximation problem, where the goal is to output an estimate $\tilde{\mu}$ of the mean $\mu = \mathbb{E}[X]$ such that $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$. A first attempt is to replace the Bernoulli estimator with the variable-time Bernoulli estimator in the algorithms of Chapter 4. Unfortunately, the latter estimator satisfies a less general deviation bound than the former one. We solve this issue by using a “weaker” version of the sub-Gaussian estimator, which increases the complexity by a factor of $\epsilon^{-1/2}$. We now sketch the construction of this estimator. Suppose we are given an upper bound $\mathcal{V} \geq \sigma/\mu$ on the coefficient of variation of a *non-negative* random variable X with stopping time T . By the same arguments as in Section 4.1.3 and Theorem 4.4.2, one can show that for the truncation level $b = \frac{2\mu(\mathcal{V}+1)}{\epsilon}$ the expectation of the random variable $Y = X \mathbb{1}_{X \leq b}$ satisfies $|\mathbb{E}[Y] - \mu| \leq \epsilon\mu/2$ (Lemma 5.4.3). Thus, by the triangle inequality, it suffices to estimate $\mathbb{E}[Y]$ with relative error $\epsilon/2$. By using the variable-time Bernoulli estimator, and given our choice of b , the expectation of Y can be estimated with a time complexity on the order of $\mathcal{V}\sqrt{\mathbb{E}[T^2]}\epsilon^{-2}$. The remaining obstacle is to find the truncation level b (or an approximation of it). For that, we define the sequence of truncation levels $b_\ell = 2^{-\ell}(\mathcal{V} + 1)$ for $\ell \geq 1$ and we estimate the mean $\mu_\ell = \mathbb{E}[X \mathbb{1}_{X \leq b_\ell}]$ by using the variable-time Bernoulli estimator for increasing values of ℓ . We set the time complexity of each estimation to $\mathcal{V}\sqrt{\mathbb{E}[T^2]}$. The crucial property is that the obtained estimates $\tilde{\mu}_\ell$ are smaller than $2^{-\ell}$ for small values of ℓ , and they become larger than $2^{-\ell}$ when $\ell \approx \log(1/\mu)$. The first part of this property stems from an analog result to Markov’s inequality, which shows that the output of the variable-time Bernoulli estimator is smaller than twice the estimated mean with high probability (Proposition 5.4.2). The second part is similar to the argument presented before, namely when $b_\ell \approx \mu(\mathcal{V} + 1)$ the variable-time Bernoulli estimator outputs an accurate estimate of μ after only $\mathcal{V}\sqrt{\mathbb{E}[T^2]}$ steps of computation. Thus, by stopping the above iterations when $\tilde{\mu}_\ell$ becomes larger than $2^{-\ell}$, we can take $b = b_\ell/\epsilon$ as an approximation of the desired truncation level.

5.2 Model of input

We assume the existence of an abstract measure of complexity, called the *time complexity*, that characterizes the total amount of time needed to execute a given quantum algorithm. This measure assigns a 0/1 cost to each gate occurring in a quantum circuit.

Assumption 5.A (TIME COMPLEXITY). Consider a fixed set \mathcal{S} of quantum gates, and a function $T_{\max} : \mathcal{S} \rightarrow \{0, 1\}$ where $T_{\max}(\mathcal{G})$ is called the *time complexity* of the gate $\mathcal{G} \in \mathcal{S}$. Given a quantum circuit \mathcal{A} over the gate set \mathcal{S} , we define the *time complexity* $T_{\max}(\mathcal{A})$ of \mathcal{A} to be the sum of the time complexities of all the gates present in \mathcal{A} .

The time complexity is intended to measure the *worst-case* cost of an algorithm. We present a second measure of complexity for characterizing the *average* cost, which is based on the definition of a *variable-time algorithm* [Amb10c; CKS17; CGJ19]. A variable-time algorithm is a product $U = U_n \cdots U_1$ of n unitaries acting on a space $\mathbb{C}^{2^n} \otimes \mathcal{H}$, where the first n registers of the memory contain binary flags that indicate whether the computation is finished or not. Each sub-algorithm U_i can only modify the states where the computation is unfinished (which corresponds to the first $i - 1$ -th flags being set to 0). The probability of the “branches of computation” that are stopped at the i -th step is denoted by $p_{\text{stop},i}$. The definition is illustrated in Figure 5.1.

Definition 5.2.1 (VARIABLE-TIME ALGORITHM). Let n be an integer and \mathcal{H} a Hilbert space. For each $i \in \{0, \dots, n\}$, define the projection operators $\Pi_{\text{stop},i} = |0^{i-1}1\rangle\langle 0^{i-1}1| \otimes I$ and $\Pi_{\text{stop},\leq i} = \sum_{j=1}^i \Pi_{\text{stop},j}$ acting on \mathbb{C}^{2^n} . We say that a quantum unitary algorithm U acting on $\mathbb{C}^{2^n} \otimes \mathcal{H}$ is a *variable-time algorithm with intermediate stopping times* $(t_i)_{i \in [n]}$ and *stopping probabilities* $(p_{\text{stop},i})_{i \in [n]}$ if it can be decomposed as a product $U = U_n \cdots U_1$ of n unitary quantum algorithms that satisfy the following conditions. For each $i \in [n]$,

- (1) There exists a unitary operator V_i acting on $\mathbb{C}^{2^{n-i+1}} \otimes \mathcal{H}$ such that

$$U_i = |0^{i-1}\rangle\langle 0^{i-1}| \otimes V_i + \Pi_{\text{stop},\leq i-1} \otimes I.$$

- (2) The time complexity of U_i is $T_{\max}(U_i) = t_i - t_{i-1}$ (where $t_0 = 0$).
- (3) The stopping probability is $p_{\text{stop},i} = \|\Pi_{\text{stop},i}(U_i \cdots U_1|\mathbf{0}\rangle)\|^2$.

Additionally, we require that $\|\Pi_{\text{stop},\leq n}U|\mathbf{0}\rangle\| = 1$.

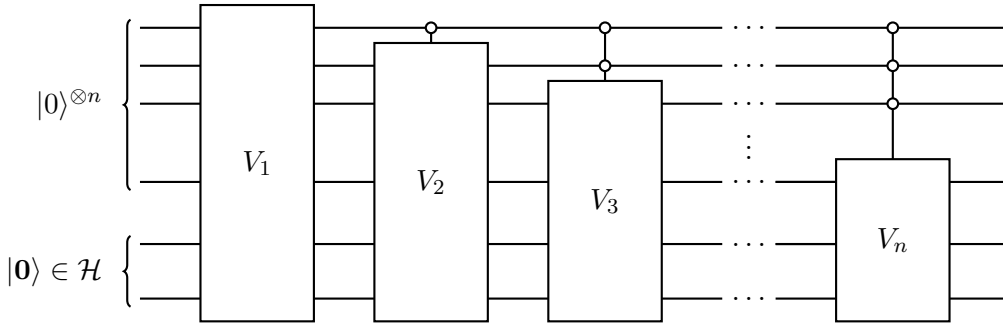


Figure 5.1: An illustration of the n steps of a variable-time algorithm $U = U_n \cdots U_1$. The i -th layer of the circuit represents the unitary operator U_i . Each V_i is controlled on the first $i - 1$ registers being equal to 0 (represented by the white circles).

The *stopping time* T of a variable-time algorithm is the random variable taking values in $\{t_1, \dots, t_n\}$ that is distributed according to the stopping probabilities of the algorithm U . The time complexity mentioned in Assumption 5.A is simply equal to the maximum value t_n taken by T . More interestingly, we can define two measures of average stopping time T_1 and T_2 by taking the expectation of T in ℓ_1 or ℓ_2 norms respectively.

Definition 5.2.2 (STOPPING TIME). The *stopping time* of a variable-time algorithm U with intermediate stopping times $(t_i)_{i \in [n]}$ and stopping probabilities $(p_{\text{stop},i})_{i \in [n]}$ is the random variable T that takes value t_i with probability $p_{\text{stop},i}$. The ℓ_1 -average stopping time is $T_1(U) = \mathbb{E}[T]$. The ℓ_2 -average stopping time is $T_2(U) = \sqrt{\mathbb{E}[T^2]}$.

We have $T_1(U) \leq T_2(U) \leq T_{\max}(U)$. We can easily adapt the definition of a q -random variable (Section 4.2) to the case where it is generated by a variable-time algorithm.

Definition 5.2.3 (STOPPING TIME OF A Q -RANDOM VARIABLE). A random variable X is a q -random variable with stopping time T if it is generated by some q -variable (\mathcal{H}, U, M) where U has stopping time T .

The comparison oracle from Assumption 4.A is not needed in this chapter. We assume that the time complexity of the rotation oracle $R_{a,b}$ defined in Assumption 4.B is at most 1.

5.3 Variable-time amplitude estimation

The main contribution of this section is the next *variable-time amplitude estimation* (VTAE) algorithm for estimating the squared amplitude $p = \|\Pi U|\mathbf{0}\rangle\|^2$ with relative error ϵ , where Π is a projector and U is a variable-time algorithm with stopping time T . The time complexity of the VTAE algorithm is on the order of $T_2(U)/(\epsilon^{3/2}\sqrt{p})$, whereas the standard amplitude estimation algorithm (Theorem 3.3.1) has time complexity $T_{\max}(U)/(\epsilon\sqrt{p})$.

Theorem 5.3.1 (VARIABLE-TIME AMPLITUDE ESTIMATION). *Let $U = U_n \cdots U_1$ be a variable-time algorithm on $\mathbb{C}^{2^n} \otimes \mathcal{H}$, for some Hilbert space \mathcal{H} , with stopping time T . Given a projection operator Π on \mathcal{H} , define the number $p \in [0, 1]$ such that $p = \|(I \otimes \Pi)U|\mathbf{0}\rangle\|^2$. Then, given two time parameters $t, T_2 \geq 1$ and two reals $\epsilon, \delta \in (0, 1)$, the variable-time amplitude estimation algorithm $\text{VT-AEst}(U, \Pi, t, T_2, \epsilon, \delta)$ outputs an estimate \tilde{p} such that*

- (1) *If $t \geq \frac{1}{\epsilon\sqrt{p}}$ and $T_2 \geq T_2(U)$ then $\Pr[|\tilde{p} - p| > \epsilon p] \leq \delta$.*
- (2) $\Pr[\tilde{p} \leq 2p] \geq 1 - \delta$.

The time complexity of the algorithm is $\tilde{O}\left(tT_2 \cdot \min\left(\frac{\log^4(T_{\max})}{\sqrt{\epsilon}}, 1 + \frac{T_{\max}}{\epsilon t T_2}\right) \log\left(\frac{1}{\delta}\right)\right)$.

The rest of this section is dedicated to the proof of this theorem. We first introduce some notations in Section 5.3.1. Next, we present in Section 5.3.2 the two state generations algorithms $(\mathcal{B}_i)_i$ and $(\mathcal{A}_i)_i$ used by Ambaini [Amb10c] for the VTAA algorithm. We make a crucial observation about the amplitude of the states they generate in Lemma 5.3.3. Finally, we use these algorithms in Section 5.3.3 to construct the VTAE algorithm.

5.3.1 Notations

For clarity in the proof, we assume the existence of an extra flag register containing a value in $\{0, 1, 2\}$ that indicates if the computation is unfinished (value 2), if it is finished and corresponds to the “accepted” part lying in the support of Π (value 1), or if it is finished and corresponds to the “rejected” part lying in the support of $I - \Pi$ (value 0). More formally, the projection operator Π is assumed to be $I \otimes |1\rangle\langle 1|$, the initial state is $|\psi^{(0)}\rangle = |\mathbf{0}\rangle|2\rangle$, and each intermediate state $|\psi^{(i)}\rangle = U_i \cdots U_1 |\psi^{(0)}\rangle$ of the variable-time algorithm $U = U_n \cdots U_1$ is written as

$$|\psi^{(i)}\rangle = \sqrt{p_{\text{rej}, \leq i}} |\psi_0^{(i)}\rangle |0\rangle + \sqrt{p_{\text{acc}, \leq i}} |\psi_1^{(i)}\rangle |1\rangle + \sqrt{p_{\text{stop}, > i}} |\psi_2^{(i)}\rangle |2\rangle$$

for some numbers $p_{\text{rej}, \leq i}$, $p_{\text{acc}, \leq i}$, $p_{\text{stop}, > i}$, and some unit states $|\psi_0^{(i)}\rangle$, $|\psi_1^{(i)}\rangle$, $|\psi_2^{(i)}\rangle$ such that $|\psi_2^{(i)}\rangle |2\rangle = (|0^i\rangle\langle 0^i| \otimes I) |\psi^{(i)}\rangle$. We have that $p_{\text{stop}, > n} = 0$ since all the computations

must be finished at step n . The quantity to be estimated is $p = \|(I \otimes |1\rangle\langle 1|)|\psi^{(n)}\rangle\|^2$, which is equal to

$$p = p_{\text{acc}, \leq n}.$$

Finally, we define the following two projectors on $\mathbb{C}^{2^n} \otimes \mathcal{H}$:

$$\begin{cases} \Pi_1 = I \otimes |1\rangle\langle 1| & (\text{projection on the accepted part}) \\ \Pi_{1,2} = I \otimes (|1\rangle\langle 1| + |2\rangle\langle 2|) & (\text{projection on the non-rejected part}). \end{cases}$$

5.3.2 State generation algorithms

We recall the definition of the two sequences of state generation algorithms $(\mathcal{B}_i)_i$ (Algorithm 5.2) and $(\mathcal{A}_i)_i$ (Algorithm 5.3) defined by Ambainis [Amb10c] in the context of VTAA. These algorithms alternate between running one of the sub-algorithms U_i , and amplifying the non-rejected part of the state lying in the support of $\Pi_{1,2}$. They take as input a sequence of amplitude estimates $(\tilde{b}_i)_i$ of $b_i = \|\Pi_{1,2}(\mathcal{B}_i|\psi^{(0)})\|^2$ that will be computed later on. The main properties of these algorithms are given in Proposition 5.3.2. The proof of parts (2) and (3) appears implicitly in [Amb10c], we give it with new details.

1. If $i = 1$, output $\mathcal{B}_1 = U_1$.
2. If $i > 1$, output $\mathcal{B}_i = U_i \mathcal{A}_{i-1}$ where $\mathcal{A}_{i-1} = \text{Gen}_{\mathcal{A}}(U, i-1, (\tilde{b}_j)_{1 \leq j \leq i-1})$.

Algorithm 5.2: State generation algorithm $\mathcal{B}_i = \text{Gen}_{\mathcal{B}}(U, i, (\tilde{b}_j)_{1 \leq j \leq i-1})$.

1. Let $\mathcal{B}_i = \text{Gen}_{\mathcal{B}}(U, i, (\tilde{b}_j)_{1 \leq j \leq i-1})$.
2. If $\tilde{b}_i > \frac{1}{9n}$, output $\mathcal{A}_i = \mathcal{B}_i$.
3. If $\tilde{b}_i \leq \frac{1}{9n}$, output the **amplitude amplification** algorithm $\mathcal{A}_i = \text{AAmp}(\mathcal{B}_i, \Pi_{1,2}, k_i)$ where k_i is the smallest integer satisfying $1/(9n) \leq (2k_i + 1)^2 \tilde{b}_i \leq 1/n$.

Algorithm 5.3: State generation algorithm $\mathcal{A}_i = \text{Gen}_{\mathcal{A}}(U, i, (\tilde{b}_j)_{1 \leq j \leq i})$.

Proposition 5.3.2. *Let U be a variable-time algorithm with intermediate stopping times $(t_i)_{i \in [n]}$ under the notations of Section 5.3.1. Given a sequence of estimates $(\tilde{b}_j)_{1 \leq j \leq n}$ define $\mathcal{B}_i = \text{Gen}_{\mathcal{B}}(U, i, (\tilde{b}_j)_{1 \leq j < i})$ (Algorithm 5.2) and $\mathcal{A}_i = \text{Gen}_{\mathcal{A}}(U, i, (\tilde{b}_j)_{1 \leq j \leq i})$ (Algorithm 5.3) for each $i \in [n]$. Let $b_i = \|\Pi_{1,2}(\mathcal{B}_i|\psi^{(0)})\|^2$ and $a_i = \|\Pi_{1,2}(\mathcal{A}_i|\psi^{(0)})\|^2$. Then, for all $i \in [n]$ and some universal constant C ,*

- (1) $b_i = a_{i-1} \frac{1 - p_{\text{rej}, \leq i}}{1 - p_{\text{rej}, \leq i-1}}$ where $a_0 = 0$.
- (2) If $|\tilde{b}_j - b_j| \leq \frac{b_j}{3n}$ for all $j \in [i]$ then $T_{\max}(\mathcal{A}_i) \leq C\sqrt{n} \left(t_i + i \frac{T_2(U)}{\sqrt{1 - p_{\text{rej}, \leq i}}} \right)$.
- (3) If $|\tilde{b}_j - b_j| \leq \frac{b_j}{3n}$ for all $j \in [i]$ then $a_i \geq (1 - \frac{1}{3n}) \frac{1}{9n}$.

Proof of part (1). The state $\mathcal{B}_1|\psi^{(0)}\rangle$ is $U_1|\psi^{(0)}\rangle = \sqrt{p_{\text{rej},\leq 1}}|\psi_0^{(1)}\rangle|0\rangle + \sqrt{p_{\text{acc},\leq 1}}|\psi_1^{(1)}\rangle|1\rangle + \sqrt{p_{\text{stop},>1}}|\psi_2^{(1)}\rangle|2\rangle$, thus $b_1 = p_{\text{acc},\leq 1} + p_{\text{stop},>1} = 1 - p_{\text{rej},\leq 1}$. For $i > 1$, the state $\mathcal{A}_{i-1}|\psi^{(0)}\rangle$ is equal to

$$\sqrt{1 - a_{i-1}}|\psi_0^{(i-1)}\rangle|0\rangle + \sqrt{a_{i-1}}\left(\sqrt{\frac{p_{\text{acc},\leq i-1}}{1 - p_{\text{rej},\leq i-1}}}|\psi_1^{(i-1)}\rangle|1\rangle + \sqrt{\frac{p_{\text{stop},> i-1}}{1 - p_{\text{rej},\leq i-1}}}|\psi_2^{(i-1)}\rangle|2\rangle\right).$$

Furthermore, there exist some unit states $|\psi_{2\rightarrow 0}^{(i-1)}\rangle$, $|\psi_{2\rightarrow 1}^{(i-1)}\rangle$, $|\psi_{2\rightarrow 2}^{(i-1)}\rangle$ such that

$$|\psi_2^{(i-1)}\rangle = \frac{1}{\sqrt{1 - p_{\text{stop},> i-1}}}\left(\sqrt{p_{\text{rej},i}}|\psi_{2\rightarrow 0}^{(i-1)}\rangle + \sqrt{p_{\text{acc},i}}|\psi_{2\rightarrow 1}^{(i-1)}\rangle + \sqrt{p_{\text{stop},> i}}|\psi_{2\rightarrow 2}^{(i-1)}\rangle\right)$$

where $p_{\text{rej},i} = p_{\text{rej},\leq i} - p_{\text{rej},\leq i-1}$, $p_{\text{acc},i} = p_{\text{acc},\leq i} - p_{\text{acc},\leq i-1}$, and $U_i(|\psi_{2\rightarrow b}^{(i-1)}\rangle|2\rangle)$ lies in the support of $I \otimes |f\rangle\langle f|$ for each $f \in \{0, 1, 2\}$. We obtain that $b_i = \|\Pi_{1,2}(\mathcal{B}_i|\psi^{(0)}\rangle)\|^2 = \|\Pi_{1,2}(U_i\mathcal{A}_{i-1}|\psi^{(0)}\rangle)\|^2 = a_{i-1}\left(\frac{p_{\text{acc},\leq i-1} + p_{\text{acc},i}}{1 - p_{\text{rej},\leq i-1}} + \frac{p_{\text{stop},> i}}{1 - p_{\text{rej},\leq i-1}}\right) = a_{i-1}\frac{1 - p_{\text{rej},\leq i}}{1 - p_{\text{rej},\leq i-1}}$. \square

Proof of parts (2) and (3). Assume that $|\tilde{b}_j - b_j| \leq b_j/(3n)$ for all $j \in [i]$. The time complexity of \mathcal{B}_i is $T_{\max}(\mathcal{B}_i) = T_{\max}(\mathcal{A}_{i-1}) + (t_i - t_{i-1})$. If $\tilde{b}_i > 1/(9n)$ then $T_{\max}(\mathcal{A}_i) = T_{\max}(\mathcal{B}_i) = T_{\max}(\mathcal{A}_{i-1}) + (t_i - t_{i-1})$ and $a_i = b_i \geq (1 + \frac{1}{3n})^{-1} \frac{1}{9n} \geq (1 - \frac{1}{3n}) \frac{1}{9n}$. If $\tilde{b}_i \leq 1/(9n)$ then by Theorem 3.2.1 the time complexity of \mathcal{A}_i is $T_{\max}(\mathcal{A}_i) = (2k_i + 1)T_{\max}(\mathcal{B}_i) = (2k_i + 1)(T_{\max}(\mathcal{A}_{i-1}) + (t_i - t_{i-1}))$. Moreover, by Corollary 3.2.2, we have $a_i \geq \left(1 - \frac{(2k_i + 1)^2 \tilde{b}_i}{3}\right)(2k_i + 1)^2 \tilde{b}_i \geq \max\left((1 - \frac{1}{3n}) \frac{1}{9n}, (1 - \frac{1}{3n})^2 (2k_i + 1)^2 b_i\right)$ where the last inequality uses the definition of k_i and the hypothesis $\tilde{b}_i \geq (1 - \frac{1}{3n})b_i$. Consequently, $T_{\max}(\mathcal{A}_i) \leq \left(1 + \frac{1}{3n-1}\right)\sqrt{\frac{a_i}{b_i}}(T_{\max}(\mathcal{A}_{i-1}) + (t_i - t_{i-1}))$. Applying this result recursively,

$$\begin{aligned} T_{\max}(\mathcal{A}_i) &\leq \left(1 + \frac{1}{3n-1}\right) \sum_{\ell=1}^i \left(\prod_{j=\ell}^i \sqrt{\frac{a_j}{b_j}}\right) (t_\ell - t_{\ell-1}) \\ &= \left(1 + \frac{1}{3n-1}\right) \sum_{\ell=1}^i \sqrt{\frac{a_i}{a_{\ell-1}}} \cdot \frac{1 - p_{\text{rej},\leq \ell-1}}{1 - p_{\text{rej},\leq i}} \cdot (t_\ell - t_{\ell-1}) \quad (\text{by part (1)}) \\ &\leq C\sqrt{n} \sum_{\ell=1}^i \sqrt{\frac{1 - p_{\text{rej},\leq \ell-1}}{1 - p_{\text{rej},\leq i}}} \cdot (t_\ell - t_{\ell-1}) \\ &\leq C\sqrt{n} \sum_{\ell=1}^i \sqrt{1 + \frac{p_{\text{stop},> \ell-1}}{1 - p_{\text{rej},\leq i}}} \cdot (t_\ell - t_{\ell-1}) \\ &\quad (\text{since } 1 - p_{\text{rej},\leq \ell-1} = p_{\text{acc},\leq \ell-1} + p_{\text{stop},> \ell-1}) \\ &\leq C\sqrt{n} \left(t_i + \frac{1}{\sqrt{1 - p_{\text{rej},\leq i}}} \sum_{\ell=1}^i \sqrt{p_{\text{stop},> \ell-1} \cdot t_\ell^2}\right) \quad (\forall x \in \mathbb{R}^+ : \sqrt{1+x} \leq 1 + \sqrt{x}) \\ &\leq C\sqrt{n} \left(t_i + i \frac{T_2(U)}{\sqrt{1 - p_{\text{rej},\leq i}}}\right). \quad (\forall \ell \in [i] : \sqrt{p_{\text{stop},> \ell-1} \cdot t_\ell^2} \leq T_2(U)) \end{aligned}$$

\square

The next lemma provides a crucial formula that expresses the acceptance probability $p_{\text{acc},\leq i}$ after i steps as a telescoping product.

Lemma 5.3.3. *Using the notations of Section 5.3.1 and Proposition 5.3.2, for each $i \in [n]$, the probability $p_{\text{acc}, \leq i}$ of the accepting part in $U_i \cdots U_1 |\psi^{(0)}\rangle$ is equal to*

$$p_{\text{acc}, \leq i} = b_1 \cdot \prod_{j=2}^{i-1} \frac{b_j}{a_{j-1}} \cdot \frac{b_{i,1}}{a_{i-1}}$$

where $b_{i,1} = \|\Pi_1(\mathcal{B}_i |\psi^{(0)}\rangle)\|^2$.

Proof. The product $b_1 \cdot \prod_{j=2}^{i-1} \frac{b_j}{a_{j-1}}$ is equal to $1 - p_{\text{rej}, \leq i-1}$ by part (1) of Proposition 5.3.2. Moreover, $b_{i,1} = a_{i-1} \frac{p_{\text{acc}, \leq i}}{1 - p_{\text{rej}, \leq i-1}}$ by the same arguments as in Proposition 5.3.2. \square

5.3.3 Main algorithm

We describe the VTAE algorithm used to prove Theorem 5.3.1. We first show in Algorithm 5.4 how to approximate the intermediate acceptance probability $p_{\text{acc}, \leq i}$ at any step $i \in \{1, \dots, n\}$. The estimate $\tilde{p}_{\text{acc}, \leq i}$ is obtained by approximating each term a_j , b_j and $b_{i,1}$ appearing in the formula of Lemma 5.3.3. The estimation is performed by using the sequential amplitude estimation algorithm Seq-AEst described in Theorem 3.3.3, which is applied to the state generation algorithms $(\mathcal{B}_i)_i$ and $(\mathcal{A}_i)_i$ from Proposition 5.3.2.

1. For $j = 1, \dots, i-1$:
 - a) Set $\mathcal{B}_j = \text{Gen}_{\mathcal{B}}(U, j, (\tilde{b}_k)_{1 \leq k \leq j-1})$, compute $\tilde{b}_j = \text{Seq-AEst}(\mathcal{B}_j, \Pi_{1,2}, \frac{\epsilon}{4n}, \frac{\delta}{2n})$.
 - b) Set $\mathcal{A}_j = \text{Gen}_{\mathcal{A}}(U, j, (\tilde{b}_k)_{1 \leq k \leq j})$, compute $\tilde{a}_j = \text{Seq-AEst}(\mathcal{A}_j, \Pi_{1,2}, \frac{\epsilon}{8n}, \frac{\delta}{2n})$.
2. Set $\mathcal{B}_i = \text{Gen}_{\mathcal{B}}(U, i, (\tilde{b}_k)_{1 \leq k \leq i-1})$, compute $\tilde{b}_{i,1} = \text{Seq-AEst}(\mathcal{B}_i, \Pi_1, \frac{\epsilon}{4n}, \frac{\delta}{2n})$.
3. Output $\tilde{p}_{\text{acc}, \leq i} = \tilde{b}_1 \cdot \prod_{j=2}^{i-1} \frac{\tilde{b}_j}{\tilde{a}_{j-1}} \cdot \frac{\tilde{b}_{i,1}}{\tilde{a}_{i-1}}$.

Algorithm 5.4: Estimation of the intermediate acceptance probability $p_{\text{acc}, \leq i}$.

Proposition 5.3.4. *Let U be a variable-time algorithm with intermediate stopping times t_1, \dots, t_n , under the notations of Section 5.3.1 and Proposition 5.3.2. Given a step $i \in [n]$ and two reals $\epsilon, \delta \in (0, 1)$, the output $\tilde{p}_{\text{acc}, \leq i}$ of Algorithm 5.4 satisfies, with probability at least $1 - \delta$,*

$$|\tilde{p}_{\text{acc}, \leq i} - p_{\text{acc}, \leq i}| \leq \epsilon p_{\text{acc}, \leq i}.$$

Moreover, the time complexity is $O\left(\frac{n^3}{\epsilon} \sqrt{\frac{1 - p_{\text{rej}, \leq i}}{p_{\text{acc}, \leq i}}} \left(t_i + i \frac{T_2(U)}{\sqrt{1 - p_{\text{rej}, \leq i}}}\right) \log\left(\frac{n}{\delta}\right)\right)$ with probability at least $1 - \delta$.

Proof. We have that $|\tilde{b}_j - b_j| \leq \frac{\epsilon}{4n} b_j$, $|\tilde{a}_j - a_j| \leq \frac{\epsilon}{8n} a_j$ (for all $j \in [i-1]$) and $|\tilde{b}_{i,1} - b_{i,1}| \leq \frac{\epsilon}{4n} b_{i,1}$ with probability at least $(1 - \frac{\delta}{2n})^{2i-1} \geq 1 - \delta$ by Theorem 3.3.3 and a union bound. Moreover, if $|\tilde{a}_j - a_j| \leq \frac{\epsilon}{8n} a_j$ then $|\frac{1}{\tilde{a}_j} - \frac{1}{a_j}| \leq \frac{\epsilon}{4n} \frac{1}{a_j}$. Consequently, the estimate $\tilde{p}_{\text{acc}, \leq i}$ satisfies with probability at least $1 - \delta$ that,

$$\tilde{p}_{\text{acc}, \leq i} \leq \left(1 + \frac{\epsilon}{4n}\right)^{2i} b_1 \cdot \prod_{j=2}^{i-1} \frac{b_j}{a_{j-1}} \cdot \frac{b_{i,1}}{a_{i-1}} \leq \left(1 + \frac{4i}{4n} \epsilon\right) \cdot p_{\text{acc}, \leq i} \leq (1 + \epsilon) p_{\text{acc}, \leq i}$$

where the first inequality uses Lemma 5.3.3, and the second inequality uses that $1 + x \leq e^x$ for $x \in \mathbb{R}$ and $e^y - 1 \leq 2y$ for $y \in [0, 1]$. Similarly, it also satisfies that,

$$\tilde{p}_{\text{acc}, \leq i} \geq \left(1 - \frac{\epsilon}{4n}\right)^{2i} b_1 \cdot \prod_{j=2}^{i-1} \frac{b_j}{a_{j-1}} \cdot \frac{b_{i,1}}{a_{i-1}} \geq \left(1 - \frac{2i}{4n}\epsilon\right) \cdot p_{\text{acc}, \leq i} \geq (1 - \epsilon)p_{\text{acc}, \leq i}$$

where the second step is by Bernoulli's inequality.

We now analyze the total time complexity. By Theorem 3.3.3 and a union bound, with probability at least $1 - \delta$, step 1.a has time complexity $O\left(\frac{n}{\epsilon\sqrt{b_j}} T_{\max}(\mathcal{B}_j) \log\left(\frac{n}{\delta}\right)\right)$, step 1.b has time complexity $O\left(\frac{n}{\epsilon\sqrt{a_j}} T_{\max}(\mathcal{A}_j) \log\left(\frac{n}{\delta}\right)\right)$ and step 2 has time complexity $O\left(\frac{n}{\epsilon\sqrt{b_{i,1}}} T_{\max}(\mathcal{B}_i) \log\left(\frac{n}{\delta}\right)\right)$. Moreover, by definitions of $(\mathcal{B}_j)_j$ and $(\mathcal{A}_j)_j$, if $\tilde{b}_j > \frac{1}{9n}$ then $a_j = b_j$ and $T_{\max}(\mathcal{A}_j) \geq T_{\max}(\mathcal{B}_j)$, and if $\tilde{b}_j \leq \frac{1}{9n}$ then $T_{\max}(\mathcal{A}_j) = \Omega\left(\sqrt{\frac{a_j}{b_j}} T_{\max}(\mathcal{B}_j)\right)$. In both cases we get $\frac{T_{\max}(\mathcal{B}_j)}{\sqrt{b_j}} = O\left(\frac{T_{\max}(\mathcal{A}_j)}{\sqrt{a_j}}\right)$. Similarly, $\frac{T_{\max}(\mathcal{B}_i)}{\sqrt{b_{i,1}}} = O\left(\sqrt{\frac{b_i}{a_i b_{i,1}}} T_{\max}(\mathcal{A}_i)\right)$. Thus, the total time complexity is $O\left(\left(\sum_{j=1}^{i-1} \frac{n}{\epsilon\sqrt{a_j}} T_{\max}(\mathcal{A}_j) + \frac{n}{\epsilon} \sqrt{\frac{b_i}{a_i b_{i,1}}} T_{\max}(\mathcal{A}_i)\right) \log\left(\frac{n}{\delta}\right)\right) = O\left(\frac{n^3}{\epsilon} \sqrt{\frac{1-p_{\text{rej}, \leq i}}{p_{\text{acc}, \leq i}}} \left(t_i + i \frac{T_2(U)}{\sqrt{1-p_{\text{rej}, \leq i}}}\right) \log\left(\frac{n}{\delta}\right)\right)$ with probability $1 - \delta$ by Proposition 5.3.2. \square

The VTAE algorithm is finally described in Algorithm 5.5. It uses the above algorithm to obtain an amplitude estimate \tilde{p} of $p = p_{\text{acc}, \leq n}$ that satisfies the properties of Theorem 5.3.1. We cannot simply run the above result on the input $i = n$ since the time complexity would depend linearly on the maximum stopping time $t_n = T_{\max}(U)$. Instead, we show that it suffices to stop the algorithm at step $i = T_2(U)/\sqrt{\epsilon p}$ to get an ϵ error approximate of p . We make the basic assumption (also used in [Amb10c; CGJ19]) that the intermediate stopping times t_i are exponentially distributed, that is $t_i = 2^i$ for $i \in [n]$ (if it is not the case, we first reslice the circuit computing U). In particular, we have that $n = \log(T_{\max}(U))$.

1. Set $i = \min(n, \lceil \log(2\sqrt{\epsilon} t T_2) \rceil)$ and $t' = 2D \frac{n^3}{\epsilon} (t_i + i \epsilon t T_2) \log\left(\frac{n}{\delta}\right)$, where D is the constant hidden in the $O(\cdot)$ notation of Proposition 5.3.4.
2. Run Algorithm 5.4 with input $U, i, \epsilon/2, \delta$ for at most t' time steps.
 - a) If the computation has not ended after t' steps, stop it and output $\tilde{p} = 0$.
 - b) Else, output $\tilde{p} = \tilde{p}_{\text{acc}, \leq i}$, where $\tilde{p}_{\text{acc}, \leq i}$ is the result of Algorithm 5.4.

Algorithm 5.5: Variable-time amplitude estimation, VT-AEst($U, t, T_2, \epsilon, \delta$).

Theorem 5.3.1 (Restated). *Let U be a variable-time algorithm under the notations of Section 5.3.1 and Proposition 5.3.2. Then, given two time parameters $t, T_2 \geq 1$ and two reals $\epsilon, \delta \in (0, 1)$, the output \tilde{p} of Algorithm 5.5 satisfies,*

- (1) *If $t \geq \frac{1}{\epsilon\sqrt{p}}$ and $T_2 \geq T_2(U)$ then $\Pr[|\tilde{p} - p| > \epsilon p] \leq \delta$.*
- (2) *$\Pr[\tilde{p} \leq 2p] \geq 1 - \delta$.*

The time complexity of the algorithm is $O\left(t T_2 \cdot \left(n + \min\left(\frac{1}{\sqrt{\epsilon}}, \frac{T_{\max}(U)}{\epsilon t T_2}\right)\right) n^3 \log\left(\frac{n}{\delta}\right)\right)$ where $n = \log(T_{\max}(U))$.

Proof. Assume first that $t \geq \frac{1}{\epsilon\sqrt{p}}$ and $T_2 \geq T_2(U)$, and suppose that $t_i = 2^i$ for all $i \in [n]$. Then, the ℓ_2 -average stopping time of U satisfies $T_2(U) \geq \sqrt{p_{\text{stop},>i} \cdot t_i^2} = \sqrt{p_{\text{stop},>i} \cdot 2^{2i}}$. Thus, by choosing $i = \min(n, \lceil \log(2\sqrt{\epsilon}tT_2) \rceil)$, the probability of stopping after step i is at most $p_{\text{stop},>i} \leq T_2(U)^2/2^{2i} \leq \epsilon p/2$. Since the probability to be estimated is $p = p_{\text{acc},\leq n}$, we get that $p \geq p_{\text{acc},\leq i} \geq p - p_{\text{stop},>i} \geq (1 - \epsilon/2)p$ and $1 - p_{\text{rej},\leq i} \leq p + p_{\text{stop},>i} \leq (1 + \epsilon/2)p$. Thus,

$$|p_{\text{acc},\leq i} - p| \leq \epsilon p/2 \quad (5.1)$$

and $D \frac{n^3}{\epsilon} \sqrt{\frac{1-p_{\text{rej},\leq i}}{p_{\text{acc},\leq i}}} \left(t_i + i \frac{T_2(U)}{\sqrt{1-p_{\text{rej},\leq i}}} \right) \log\left(\frac{n}{\delta}\right) < t'$. Consequently, by Proposition 5.3.4, the algorithm reaches step 2.b and obtain an estimate $\tilde{p}_{\text{acc},\leq i}$ such that

$$|\tilde{p}_{\text{acc},\leq i} - p_{\text{acc},\leq i}| \leq \epsilon p_{\text{acc},\leq i}/2 \quad (5.2)$$

with probability at least $1 - \delta$. By Equations (5.1) and (5.2) and the triangle inequality, we get that $|\tilde{p}_{\text{acc},\leq i} - p| \leq \epsilon p$ with probability at least $1 - \delta$.

Assume now that $t \leq \frac{1}{\epsilon\sqrt{p}}$. According to Proposition 5.3.4, the estimate $\tilde{p}_{\text{acc},\leq i}$ obtained at step 2.b satisfies $\tilde{p}_{\text{acc},\leq i} \leq (1 + \epsilon/2)p_{\text{acc},\leq i} \leq 2p$ with probability at least $1 - \delta$. The output \tilde{p} of the algorithm is either 0 or $\tilde{p}_{\text{acc},\leq i}$, thus it satisfies $\tilde{p} \leq 2p$ with probability at least $1 - \delta$.

The time complexity is deduced from the stopping condition used at step 2. \square

5.4 Variable-time mean estimator

We use the VTAE algorithm given in the previous section to construct a series of efficient mean estimators for q -random variables generated by variable-time algorithms. We first present a variable-time Bernoulli estimator in Section 5.4.1 whose complexity depends on the largest value taken by X . We build upon this algorithm to obtain faster estimators in Section 5.4.2, whose complexities depend on the coefficient of variation of X .

5.4.1 Variable-time Bernoulli estimator

We adapt the Bernoulli estimator presented in Proposition 4.4.1 to the case of a q -random variable generated by a variable-time algorithm. We first explain how to transform the latter algorithm into a new variable-time algorithm U that encodes the expectation of the truncated random variable as an amplitude into the state $U|0\rangle$.

Proposition 5.4.1. *Let X be a q -random variable with stopping time T , and set as input two range values $0 \leq a < b$. Then, there exist a variable-time algorithm $U = U_n \cdots U_1$ with stopping time $O(T)$ acting on some Hilbert space $\mathbb{C}^{2^n} \otimes \mathcal{H}$, and a projector Π acting on \mathcal{H} , such that*

$$\|(I \otimes \Pi)U|0\rangle\|^2 = \frac{\mathbb{E}[X \mathbb{1}_{a < X \leq b}]}{b}.$$

Proof. By Definition 5.2.3, there exists a q -variable $(\mathbb{C}^{2^n} \otimes \mathcal{H}', V, M)$ generating X , where $V = V_n \cdots V_1$ is a variable-time algorithm with stopping time T acting on $\mathbb{C}^{2^n} \otimes \mathcal{H}'$ for some Hilbert space \mathcal{H}' . Let $\mathcal{H} = \mathcal{H}' \otimes \mathbb{C}^4$. Given the projector $\Pi_{\text{stop},i} = |0^{i-1}1\rangle\langle 0^{i-1}1| \otimes I$ acting on \mathbb{C}^{2^n} (Definition 5.2.1), we define the unitary $U_{\text{stop},i} = \Pi_{\text{stop},i} \otimes I \otimes X + (I - \Pi_{\text{stop},i}) \otimes I$, acting on $\mathbb{C}^{2^n} \otimes \mathcal{H}$, that flips the value of the last qubit if the flag registers indicate that the computation is stopped at the i -th step.

Consider the controlled rotation $R_{a,b}$ acting on $\mathbb{C}^{2^n} \otimes \mathcal{H}' \otimes \mathbb{C}^2$ (Assumption 4.B). We modify the variable-time algorithm V so that, after each step i , it applies $R_{a,b}$ to the branches of computation that are just finished. This is formalized by defining the new variable-time algorithm $U = U_n \cdots U_1$ defined as,

$$U_i = (R_{a,b} \otimes |1\rangle\langle 1| + I \otimes |0\rangle\langle 0|) \cdot U_{\text{stop},i} \cdot (V_i \otimes I).$$

The operators $(R_{a,b} \otimes |1\rangle\langle 1| + I \otimes |0\rangle\langle 0|) \cdot U_{\text{stop},i}$ and $V_j \otimes I$ commute when $i < j$ since they act as non-identities on disjoint subspaces by Definition 5.2.1. Moreover, $\|(I \otimes |0\rangle\langle 0|)U|\mathbf{0}\rangle\| = 0$ since all the basis states in the support of $V|\mathbf{0}\rangle$ must contain a flag register with a 1 (all the branches of computation are finished at the end). Using these two facts, we obtain that the final state $U|\mathbf{0}\rangle$ is equal to the state $(R_{a,b}(V \otimes I_2)|\mathbf{0}\rangle)|1\rangle$ obtained by applying $R_{a,b}$ only once at the end of V . In order to simplify the analysis of the latter state, let us assume that the random variable X is distributed in (a, b) . Then, $\mu = \mathbb{E}[X] = \mathbb{E}[X \mathbb{1}_{a < X \leq b}]$ and,

$$\begin{aligned} U|\mathbf{0}\rangle &= (R_{a,b} \otimes I_2) \sum_{\omega \in \Omega} \sqrt{p(\omega)} |\omega\rangle |01\rangle && \text{by Definition 5.2.3} \\ &= \sum_{\omega \in \Omega} \sqrt{p(\omega)} |\omega\rangle \left(\sqrt{1 - \frac{X(\omega)}{b}} |0\rangle + \sqrt{\frac{X(\omega)}{b}} |1\rangle \right) |1\rangle && \text{by Assumption 4.B} \\ &= \sqrt{1 - \frac{\mu}{b}} \left(\sum_{\omega \in \Omega} \sqrt{\frac{p(\omega)(b - X(\omega))}{b - \mu}} |\omega\rangle \right) |01\rangle + \sqrt{\frac{\mu}{b}} \left(\sum_{\omega \in \Omega} \sqrt{\frac{p(\omega)X(\omega)}{\mu}} |\omega\rangle \right) |11\rangle. \end{aligned}$$

Thus, $\frac{\mu}{b} = \|(I \otimes \Pi)U|\mathbf{0}\rangle\|^2$ where $\Pi = |11\rangle\langle 11|$. This result also holds if X takes values outside the interval $(a, b]$ (it only changes the unit states in front of $|01\rangle$ and $|11\rangle$). The stopping time of U is $O(T)$ since $T_{\max}(U_i) \leq O(T_{\max}(V_i))$ for all $i \in [n]$. \square

We estimate the expectation of the truncated random variable $X \mathbb{1}_{a < X \leq b}$ by using the VTAE algorithm on the variable-time algorithm constructed in the above proposition.

Proposition 5.4.2 (VARIABLE-TIME BERNOULLI ESTIMATOR). *There is a quantum algorithm, called the variable-time Bernoulli estimator, with the following properties. Let X be a q -random variable distributed in $[0, 1]$ with stopping time T , and set as input two time parameters $t, T_2 > 1$, two range values $0 \leq a < b$, and two reals $\epsilon, \delta \in (0, 1)$. Then, the variable-time Bernoulli estimator $\text{VT-BernEst}(X, t, T_2, a, b, \epsilon, \delta)$ outputs an estimate $\tilde{\mu}_{a,b}$ of $\mu_{a,b} = \mathbb{E}[X \mathbb{1}_{a < X \leq b}]$ such that,*

- (1) If $t \geq \frac{\sqrt{b}}{\epsilon \sqrt{\mu_{a,b}}}$ and $T_2 \geq \sqrt{\mathbb{E}[T^2]}$ then $\Pr[|\tilde{\mu}_{a,b} - \mu_{a,b}| > \epsilon \mu_{a,b}] \leq \delta$.
- (2) $\Pr[\tilde{\mu}_{a,b} \leq 2\mu_{a,b}] \geq 1 - \delta$.

The time complexity of the estimator is $\tilde{O}\left(\frac{t}{\sqrt{\epsilon}} T_2 \cdot \log^4(T_{\max}) \log(1/\delta)\right)$.

Proof. Let U and Π denote the variable-time algorithm and the projector provided by Proposition 5.4.1 on input X, a, b . The algorithm consists of using the variable-time amplitude estimation algorithm $\text{VT-AEst}(U, \Pi, t, T_2, \epsilon, \delta)$ (Theorem 5.3.1) to obtain an estimate \tilde{v} of $\mu_{a,b}/b$, and to output $\tilde{\mu}_{a,b} = b\tilde{v}$. The result follow directly from Proposition 5.4.1 and Theorem 5.3.1 (the expression of the time complexity has been simplified). \square

As was the case with the Bernoulli estimator, we can use the variable-time Bernoulli estimator with parameters $a = 0$ and $b = 1$ to estimate the mean μ of a random variable distributed in $[0, 1]$. However, the time complexity is on the order of $\frac{T_2}{\epsilon^{3/2} \sqrt{\mu}}$, which is often sub-optimal. We describe a more efficient approach in the next section.

5.4.2 Variable-time (ϵ, δ) -estimator

We provide three estimators whose time complexity is on the order of $\frac{\sigma T_2}{\epsilon^2 \mu}$. Our main result (Theorem 5.4.4) builds upon the *relative estimator* of Corollary 4.5.3. The analysis is based on the next lemma that bounds the truncated mean at different truncation levels.

Lemma 5.4.3. *Let X be a non-negative random variable. For any numbers $\Gamma, c, m > 0$ such that $\Gamma \geq \sqrt{\mathbb{E}[X^2]}/\mathbb{E}[X]$ and $m \geq c\mathbb{E}[X]$, we have $(1 - \frac{1}{c})\mathbb{E}[X] \leq \mathbb{E}[X \mathbb{1}_{X \leq m\Gamma^2}] \leq \mathbb{E}[X]$.*

Proof. We have that $\mathbb{E}[X \mathbb{1}_{X \leq m\Gamma^2}] = \mathbb{E}[X] - \mathbb{E}[X \mathbb{1}_{X > m\Gamma^2}]$ and $0 \leq \mathbb{E}[X \mathbb{1}_{X > m\Gamma^2}] \leq \mathbb{E}[X^2 \mathbb{1}_{X > m\Gamma^2}]/(m\Gamma^2) \leq \mathbb{E}[X^2]/(m\Gamma^2) \leq (1/c) \cdot \mathbb{E}[X]$. \square

The algorithm uses two extra input parameters compared to Corollary 4.5.3: a candidate upper bound T_2 on the ℓ_2 -average stopping time, and a candidate lower bound α on the mean μ . The need for α is removed in Proposition 5.4.6 by doing an exponential search.

1. Set $m = 1/2$.
2. Set $b_m = m(\mathcal{V}^2 + 1)$. Compute an estimate $\tilde{\mu}_m$ of $\mathbb{E}[X \mathbb{1}_{X \leq b_m}]$ by using the **variable-time Bernoulli estimator** $\text{VT-BernEst}(X, t, T_2, 0, b_m, \epsilon', \delta')$ with $t = 2\sqrt{2}(\mathcal{V} + 1)$ and $\epsilon' = 1/2$, $\delta' = \frac{\delta}{4+2\log(1/u)}$.
3. If $\tilde{\mu}_m < m/16$ and $m \geq \alpha$ then set $m = m/2$ and go to step 2.
4. Set $b = \frac{m(\mathcal{V}^2 + 1)}{\epsilon}$. Compute an estimate $\tilde{\mu}$ of $\mathbb{E}[X \mathbb{1}_{X \leq b}]$ by using the **variable-time Bernoulli estimator** $\text{VT-BernEst}(X, t, T_2, 0, b, \epsilon, \delta/2)$ with $t = \frac{8(\mathcal{V} + 1)}{\epsilon^{3/2}}$. Output $\tilde{\mu}$.

Algorithm 5.6: Variable-time relative estimator, $\text{VT-RelatEst}(X, \mathcal{V}, T_2, \alpha, \epsilon, \delta)$.

Theorem 5.4.4 (VARIABLE-TIME RELATIVE ESTIMATOR). *Let X be a q -random variable distributed in $[0, 1]$ with mean μ , variance σ^2 and stopping time T . Set as input two time parameters $\mathcal{V}, T_2 \geq 1$, a lower bound $\alpha \geq 0$, and two reals $\epsilon, \delta \in (0, 1)$. Then, the mean estimate $\tilde{\mu}$ computed by the variable-time relative estimator $\text{VT-RelatEst}(X, \mathcal{V}, T_2, \alpha, \epsilon, \delta)$ (Algorithm 5.6) satisfies*

(1) *If $\mathcal{V} \geq \sigma/\mu$, $T_2 \geq \sqrt{\mathbb{E}[T^2]}$ and $\mu \geq \alpha$ then $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$.*

(2) $\Pr[\tilde{\mu} \leq 2\mu] \geq 1 - \delta$.

The time complexity of the estimator is $\tilde{O}(\mathcal{V}T_2 \cdot (\log(\frac{1}{\alpha}) + \frac{1}{\epsilon^2}) \cdot \log^4(T_{\max}) \log(\frac{1}{\delta}))$.

Proof. We prove part (1) of the theorem. Let us assume that $\mathcal{V} \geq \sigma/\mu$ and $\mu \geq \alpha$. Note that $\mathcal{V} + 1 \geq \sqrt{\mathbb{E}[X^2]}/\mathbb{E}[X]$. We make the assumption that all calls to the variable-time Bernoulli estimator satisfy parts (1) and (2) of Proposition 5.4.2, which is the case with probability at least $1 - \delta$ by a union bound. First, we show that the value m at step 4 satisfies the following constant relative error bound,

$$2\mu \leq m \leq 32\mu. \quad (5.3)$$

Let $\mu_m = \mathbb{E}[X \mathbb{1}_{X \leq b_m}]$ denote the expectation of the truncated random variable estimated at step 2 of the algorithm. If $m > 32\mu$ then $\tilde{\mu}_m \leq 2\mu_m \leq 2\mu < m/16$, where the first

inequality is by Proposition 5.4.2. If $m \in [2\mu, 4\mu]$ then $\mu_m \geq \mu/2$ by Lemma 5.4.3, and $\frac{\sqrt{b_m}}{\sqrt{\mu_m}} \leq 2\sqrt{2}(\mathcal{V} + 1)$. Thus, by Proposition 5.4.2, if $m \in [2\mu, 4\mu]$ then $|\tilde{\mu}_m - \mu_m| \leq \mu_m/2$, and in particular $\tilde{\mu}_m \geq \mu/4 \geq m/16$. Consequently, the algorithm reaches step 4 with a value m that satisfies Equation (5.3). Next, we show that the computation at step 4 has the effect of decreasing the relative error to ϵ . Let $\mu_b = \mathbb{E}[X \mathbb{1}_{X \leq b}]$ denote the expectation estimated at step 4. Since $m \geq 2\mu$ the threshold value $b = \frac{m(\mathcal{V}^2 + 1)}{\epsilon}$ is at least

$$b \geq \frac{2\mathbb{E}[X^2]}{\epsilon\mathbb{E}[X]}. \quad (5.4)$$

Consequently, by Lemma 5.4.3, the estimated mean μ_b satisfies $|\mu_b - \mu| \leq (\epsilon/2)\mu$. Moreover, the time parameter t used at step 4 is at least $t = \frac{8(\mathcal{V}+1)}{\epsilon^{3/2}} \geq \frac{2\sqrt{b}}{\epsilon\sqrt{\mu_b}}$ since $\mu_b \geq \mu/2$ and $b \leq \frac{32(\mathcal{V}^2+1)}{\epsilon}$. Thus, by Proposition 5.4.2, the estimate $\tilde{\mu}$ satisfies $|\tilde{\mu} - \mu_b| \leq (\epsilon/2)\mu$. By the triangle inequality, we conclude that $|\tilde{\mu} - \mu| \leq \epsilon\mu$. This proves part (1) of the theorem. Part (2) and the time complexity are directly implied by Proposition 5.4.2. \square

Similarly to Theorem 4.5.4, we consider the weaker hypothesis where, instead of a direct upper bound \mathcal{V} on σ/μ , we have a non-increasing function f such that $f(\mu) \geq \sigma/\mu$. We describe an interval estimator that decides if the mean μ lies in the interval $[\alpha, \beta]$ or is ϵ -far from it. Our approach relies on the two properties proved in the previous theorem.

1. Compute an estimate $\tilde{\mu}$ of $\mu = \mathbb{E}[X]$ by using the **variable-time relative estimator** $\text{VT-RelatEst}(X, \mathcal{V}, T_2, \alpha, \epsilon/4, \delta)$ with $\mathcal{V} = f(\frac{\alpha}{4})$.
2. If $\tilde{\mu} \in [(1 - \frac{\epsilon}{2})\alpha, (1 + \frac{\epsilon}{2})\beta]$ then output $B = 1$, else output $B = 0$.

Algorithm 5.7: Variable-time interval estimator, $\text{VT-IntervEst}(X, f, T_2, \alpha, \beta, \epsilon, \delta)$.

Proposition 5.4.5 (VARIABLE-TIME INTERVAL ESTIMATOR). *Let X be a q -random variable distributed in $[0, 1]$ with mean μ , variance σ^2 and stopping time T . Set as input a non-increasing function $f \geq 1$, a time parameter $T_2 \geq 1$, two endpoints $\alpha < \beta$, and two reals $\epsilon, \delta \in (0, 1)$. Let B denote the Boolean value computed by the variable-time interval estimator $\text{VT-IntervEst}(X, f, T_2, \alpha, \beta, \epsilon, \delta)$ (Algorithm 5.7). If $f(\mu) \geq \sigma/\mu$ and $T_2 \geq \sqrt{\mathbb{E}[T^2]}$ then*

- (1) *If $\mu \in [\alpha, \beta]$ then $B = 1$ with probability at least $1 - \delta$.*
- (2) *If $\mu \notin [(1 - \epsilon)\alpha, (1 + \epsilon)\beta]$ then $B = 0$ with probability at least $1 - \delta$.*

The time complexity of the algorithm is $\tilde{O}(f(\frac{\alpha}{4})T_2 \cdot (\log(\frac{1}{\alpha}) + \frac{1}{\epsilon^2}) \cdot \log^4(T_{\max}) \log(\frac{1}{\delta}))$.

Proof. Assume first that $\mu \geq \alpha/4$. Then, $\mathcal{V} \geq \sigma/\mu$ and, by part (1) of Theorem 5.4.4, $|\tilde{\mu} - \mu| \leq (\epsilon/4)\mu$ with probability at least $1 - \delta$. Thus, with probability at least $1 - \delta$,

- if $\mu \in [\alpha, \beta]$ then $\tilde{\mu} \in [(1 - \frac{\epsilon}{4})\alpha, (1 + \frac{\epsilon}{4})\beta]$,
- if $\mu \geq (1 + \epsilon)\beta$ then $\tilde{\mu} \geq (1 - \epsilon/4)\mu \geq (1 - \epsilon/4)(1 + \epsilon)\beta \geq (1 + \epsilon/2)\beta$,
- if $\mu \leq (1 - \epsilon)\alpha$ then $\tilde{\mu} \leq (1 + \epsilon/4)\mu \leq (1 + \epsilon/4)(1 - \epsilon)\alpha \leq (1 - \epsilon/2)\alpha$.

In all three cases, the output B is correct with probability at least $1 - \delta$. Assume now that $\mu \leq \alpha/4$. By part (2) of Theorem 5.4.4, we have $\tilde{\mu} \leq 2\mu \leq \alpha/2 \leq (1 - \epsilon/2)\alpha$ with probability at least $1 - \delta$. Thus, the output is $B = 0$ with probability at least $1 - \delta$. \square

We finally describe an estimator that removes the need for a lower bound α on μ by combining the two previous algorithms. In order to simplify the analysis, we only consider the case where f is a power function. The algorithm is similar to the exponential estimator of Corollary 4.5.6.

1. Set $\alpha = 1/2$ and $\delta' = \delta/2$.
2. Compute a Boolean value B by running the **variable-time interval estimator** $\text{VT-IntervEst}(X, f, T_2, \alpha, \beta, \epsilon', \delta')$ with $\beta = 1$ and $\epsilon' = 1/2$.
 - a) If $B = 0$ then set $\alpha = \alpha/2$, $\delta' = \delta/2$ and go to step 2.
 - b) Else, output the estimate $\tilde{\mu}$ obtained by using the **variable-time relative estimator** $\text{VT-RelatEst}(X, \mathcal{V}, T_2, \alpha/2, \epsilon, \delta'/2)$ with $\mathcal{V} = f(\alpha/2)$.

Algorithm 5.8: Variable-time exponential estimator, $\text{VT-ExpEst}(X, f, T_2, \epsilon, \delta)$.

Proposition 5.4.6 (VARIABLE-TIME EXPONENTIAL ESTIMATOR). *Let X be a q -random variable distributed in $[0, 1]$ with mean μ , variance σ^2 and stopping time T . Set as input a function $f : x \mapsto \max(1, c/x^d)$ for two constants $c, d \geq 0$ such that $f(\mu) \geq \sigma/\mu$, a time parameter $T_2 \geq \sqrt{\mathbb{E}[T^2]}$, and two reals $\epsilon, \delta \in (0, 1)$ such that $\delta < 2^{-2d}$. Then, the mean estimate $\tilde{\mu}$ computed by the variable-time exponential estimator $\text{VT-ExpEst}(X, f, T_2, \epsilon, \delta)$ (Algorithm 5.8) satisfies $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$. Moreover, the time complexity C of the estimator satisfies*

1. $\Pr\left[C > \tilde{O}\left(\frac{f(\mu)}{\epsilon^2} T_2 \cdot \log^3\left(\frac{1}{\mu}\right) \log^4(T_{\max}) \log\left(\frac{1}{\delta}\right)\right)\right] \leq \delta,$
2. $\mathbb{E}[C] \leq \tilde{O}\left(\frac{f(\mu)}{\epsilon^2} T_2 \cdot \log^3\left(\frac{1}{\mu}\right) \log^4(T_{\max}) \log\left(\frac{1}{\delta}\right)\right).$

Proof. According to Proposition 5.4.5, the probability to obtain $B = 0$ at step 2 is at least $1 - \delta'$ when $\alpha \geq 2\mu$, and at most δ' when $\alpha \leq \mu$. Consequently, the value α at step 2.b satisfies $\alpha \in [\mu/2, 2\mu]$ with probability at least $1 - \delta/2$. In this case, $\mathcal{V} \geq \sigma/\mu$ and the output $\tilde{\mu}$ satisfies $|\tilde{\mu} - \mu| \leq \epsilon\mu$ with probability at least $1 - \delta/2$.

The total number C^- of operations performed until α gets smaller than $\mu/2$ is at most $C^- \leq \tilde{O}\left(\frac{f(\mu)}{\epsilon^2} \log^3(1/\mu) T_2 \cdot \log^4(T_{\max}) \log(1/\delta)\right)$. Each time step 2 is executed with $\alpha \leq \mu/2$ the probability that it outputs 1 is at least $1 - \delta'$. Thus, the expectation of the number C^+ of operations performed while α is smaller than $\mu/2$ is

$$\begin{aligned} \mathbb{E}[C^+] &\leq \tilde{O}\left(\sum_{k=0}^{\infty} \delta^k \cdot \frac{f(2^{-k}\mu)}{\epsilon^2} \log(2^k/\mu) T_2 \cdot \log^4(T_{\max}) \log(2^k/(\mu\delta))\right) \\ &\leq \tilde{O}\left(\sum_{k=0}^{\infty} \delta^k \cdot 2^{kd} k^2 \frac{f(\mu)}{\epsilon^2} \log^2(1/\mu) T_2 \cdot \log^4(T_{\max}) \log(1/\delta)\right) \\ &\leq \tilde{O}\left(\frac{f(\mu)}{\epsilon^2} \log^2(1/\mu) T_2 \cdot \log^4(T_{\max}) \log(1/\delta)\right). \end{aligned}$$

The total time complexity is $C = C^- + C^+$. \square

5.5 Discussion

We observe that the use of the ℓ_2 -average stopping time comes at the cost of a larger dependence on the error parameter ϵ compared to the algorithms based on the maximum stopping time (as in Chapter 4). There is an overhead factor of $\epsilon^{-1/2}$ for the amplitude estimation in the variable-time setting (Theorem 3.3.1 vs. Theorem 5.3.1), and an overhead factor of ϵ^{-1} for the (ϵ, δ) -mean estimation problem (Corollary 4.5.3 vs. Theorem 5.4.4). We leave as an open problem to improve the complexities in ϵ . A related question is to find an efficient variable-time mean estimator achieving the sub-Gaussian deviation bound $\Pr[|\tilde{\mu} - \mu| > \frac{\sigma \log(1/\delta)}{t}] \leq \delta$ studied in the previous chapter. The results presented in the current chapter are tailored to the (ϵ, δ) -approximation guarantee.

6

Estimation of Graph Parameters

This chapter is based on the following papers:

[HM19] Y. Hamoudi and F. Magniez. “Quantum Chebyshev’s Inequality and Applications”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 69:1–69:16.

[HM21a] Y. Hamoudi and F. Magniez. “Quantum Approximate Triangle Counting”. In submission. 2021.

6.1 Introduction

The two previous chapters studied the mean estimation problem in the “black-box” model, where the input consists of several independent runs of a random or quantum process whose outcome encodes the distribution of an *arbitrary* random variable X . Our results characterized the number of experiments and the time complexity needed to estimate the mean μ of X with some prescribed accuracy. In this chapter, we consider a more specific version of the mean estimation problem, where the random variable X encodes some graph parameter. We show that having oracle access to the underlying graph may lead to faster mean estimation algorithms. We focus on achieving the (ϵ, δ) -approximation guarantee $\Pr[|\tilde{\mu} - \mu| > \epsilon\mu] \leq \delta$, that was studied before in Section 4.5 and Chapter 5.

The mean estimation problem takes on a different tone when the process that generates X is not arbitrary. In a seminal work, Feige [Fei06] showed for instance that the average degree in an n -vertex graph is easier to estimate than the average value of n arbitrary numbers. This result prompted the study of sublinear-time algorithms for subgraph counting problems [GR08; GRS11; ORRR12; NO08; YY12; ELRS17; ERS20b; ERS20a]. These algorithms use the graph representation in the adjacency list or adjacency matrix model to achieve faster estimations than in the black-box setting. The generic mean estimators are still of great importance here, but they must be combined with more advanced mean estimation procedures. A typical approach [GR08; GRS11; ELRS17; ERS20b] is to design a random process, based on local graph exploration techniques, that outputs a random variable X whose mean equals a parameter of the graph. The objective is to minimize both the coefficient of variation of X (that controls the number of samples needed to approximate the mean) and the graph exploration time T (that determines the average stopping time for generating one sample). In the classical setting, the expected time complexity of this method is the product $N \times T_1$ of the number N of classical samples needed and of the average exploration time $T_1 = \mathbb{E}[T]$. The quantum variable-time mean estimators constructed in Chapter 5 lead to a tradeoff on the order of $\sqrt{N} \times T_2$, with a quadratic improvement over the first part of the product, but a larger dependence on the

ℓ_2 -average exploration time $T_2 = \sqrt{\mathbb{E}[T^2]}$ for the second part. We investigate the question of whether such estimators may speed up the approximation of graph parameters.

We focus our study on the quantum query complexity of the *Triangle Counting* problem in the *general graph model* [KKR04; Gol17], where both neighbor and edge queries are permitted. This model is commonly used for handling arbitrary graphs whose sparsity is unknown. In a recent work, Eden, Levi, Ron and Seshadhri [ELRS17] showed that the classical query complexity for estimating the number t of triangles in an n -vertex with m -edges is $O^*(n/t^{1/3} + \min(m, m^{3/2}/t))$ ¹. We present an optimal quantum algorithm that achieves a quadratic speedup over this quantity when $t \geq \Omega(\sqrt{m})$. Our result builds on the classical algorithms for approximate triangle counting [ELR15; Ses15; ELRS17], and on the variable-time mean estimators constructed in Chapter 5.

6.1.1 Related work

We refer the reader to Chapters 4 and 5 for related work on the mean estimation problem in the “black-box” model. The approximation of several graph parameters has been considered in the classical sublinear time literature, such as the number of edges [Fei06; GR08], stars [GRS11], triangles [ELRS17], k -cliques [ERS20b; ERS20a] or the size of a minimum spanning tree [CRT05] or a maximum matching [NO08; YY12]. These results have broad applications in diverse areas, such as social network analysis or bioinformatics. None of these problems has been studied in the quantum setting to our knowledge. There are a few quantum algorithms for estimating other graph parameters, such as electrical network quantities [IJ19; JJKP18; Wan17; Pid19; CGJ19; AW20], the circuit rank of a graph [DKW19], or the number of colorings, matchings or independent sets [Mon15; HW20]. There is a much more extensive literature on quantum algorithms for deciding graph properties [BDF+04; SYZ04; DHHM06; CK12], especially for detecting the presence of a subgraph in a graph [MSS07; CK12; LMS12; Zhu12; BR12; LMS17]. In particular, the *Triangle Finding* problem has received a great deal of interest [MSS07; Bel12; Gal14; CLM20], and determining its query complexity is still an open problem. Some recent results in fine-grained complexity have drawn connections between subgraph counting and subgraph finding problems [DL18; DLM20], but it is unclear how strong the link is [FGP20; ERR20]. Our algorithm for triangle counting is rather different than the previous quantum algorithms for triangle finding, which have been developed mainly in the quantum walk framework. However, it shall be noted that the *variable-time amplitude amplification* algorithm has played a role in the Triangle Finding problem [Gal14; GN17], whereas our result is based on the *variable-time amplitude estimation* algorithm.

6.1.2 Contributions and organization

We define the input model and the graph terminology in Section 6.2. Next, we present an optimal quantum algorithm for estimating the number of edges in a graph in Section 6.3. We obtain the following result that is needed for our triangle counting algorithm.

Theorem 6.3.2 (Restated). *There exists a quantum algorithm with the following properties. Let G be a graph with n vertices and m edges in the quantum adjacency list model, and fix two parameters $\epsilon, \delta \in (0, 1/2)$. Then, the algorithm outputs an edge estimate \tilde{m} such that $|\tilde{m} - m| \leq \epsilon m$ with probability at least $1 - \delta$. The expected quantum query complexity of the algorithm is $O^*\left(\frac{\sqrt{n}}{m^{1/4}}\right)$.*

¹In this chapter, we use the notation $\tilde{O}(x)$ to hide any polynomial factor in $\log(x)$, and the notation $O^*(x)$ to hide any polynomial factor in $\log(x)$, $\log(n)$, $\log(1/\delta)$ and $1/\epsilon$.

The quantum triangle counting algorithm is presented in Section 6.4. It relies on a series of assumptions described in Section 6.4.1, that we remove later on. The main concepts used in the algorithm are introduced in Section 6.4.2. The algorithm itself is described in Sections 6.4.3–6.4.5. The final result (Algorithm 6.9) achieves the following complexity.

Theorem 6.4.18 (Restated). *There exists a quantum algorithm with the following properties. Let G be a graph with n vertices, m edges and t triangles in the quantum general graph model, and fix an error parameter $\epsilon \in (0, 1/2)$. Then, the algorithm outputs a triangle estimate \tilde{t} such that $|\tilde{t} - t| \leq \epsilon t$ and it performs $O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$ quantum queries with probability at least $1 - 1/\log(n)$.*

We prove a lower bound for the Edge Counting problem in Proposition 6.5.3 that matches the complexity of our algorithm. We also obtain the next result for Triangle Counting that is optimal (up to logarithmic factors) when $t \geq \Omega(\sqrt{m})$.

Proposition 6.5.5 (Restated). *Any algorithm that estimates the number t of triangles in an n -vertex m -edge graph with relative error $\epsilon = 1/2$ and success probability $2/3$ must perform at least $\tilde{\Omega}\left(\frac{\sqrt{n}}{t^{1/6}} + \min\left(\frac{m^{3/4}}{\sqrt{t}}, \sqrt{m}\right)\right)$ quantum queries in the general graph model.*

6.1.3 Proof overview

We present a high overview of the triangle counting algorithm and we explain how it differs from previous classical work. Our result builds on a series of classical sublinear algorithms [ELR15; Ses15; ELRS17] and it uses the variable-time quantum mean estimators developed in Chapter 5. We assume for most of the presentation that the algorithm has prior knowledge of an edge estimate $\bar{m} \in [m/4, m]$ (Assumption 6.A) and a triangle estimate $\bar{t} \in [t/8, t]$ (Assumption 6.B). The purpose of the algorithm is to obtain a finer estimate $\tilde{t} \in [(1 - \epsilon)t, (1 + \epsilon)t]$ given a fixed error parameter $\epsilon \in (0, 1)$. We explain later on how to get rid of these assumptions by estimating the edge count m with a separate algorithm (Section 6.3), and by running the triangle counting algorithm over a decreasing sequence of values for \bar{t} (Algorithm 6.9).

The core of the algorithm is made of four estimators summarized in Table 6.1. In each of the four cases, the expectation μ of the estimator equals some quantity related to the total triangle count and we know a non-increasing function f such that the coefficient of variation σ/μ is upper bounded by $f(\mu)$. The variable-time quantum mean estimators developed in Chapter 5 (Theorem 5.4.4 and Proposition 5.4.6) can estimate the expectation μ of any such estimator in time roughly $O^*(f(\mu)T_2)$, where T_2 is an upper bound on the ℓ_2 -average stopping time of the considered estimator. Alternatively, the variable-time interval estimator (Proposition 5.4.5) can decide if the mean μ is above some threshold value α in time $O^*(f(\alpha)T_2)$. Below, we describe the estimators of Table 6.1 and we explain how they can be used to solve the Triangle Counting problem.

Buckets partitioning. Our starting point is to consider a discretization of the interval $[1, n^2]$ into a sequence $1 = \nu_0 < \nu_1 < \dots < \nu_k = n^2$ of $O^*(\log n)$ values, where each two consecutive numbers differ by a small factor on the order of $1 + \epsilon$. These values define a partition of the graph vertices into k buckets B_1, \dots, B_k , where each bucket B_i consists of all the vertices whose triangle-degree t_v lies between ν_i and ν_{i+1} (Assumption 6.D). If we could compute an estimate \tilde{s}_i of each bucket size $|B_i|$ with relative error $O(\epsilon)$, then the triangle estimate $\tilde{t} = \frac{1}{3} \sum_{i \in [k]} \tilde{s}_i \nu_i$ would solve the Triangle Counting problem. The most natural algorithm for computing \tilde{s}_i is to estimate the expectation of the unbiased

Estimator	Expectation	Coefficient of variation	ℓ_2 -average stopping time
Triangle-degree T_v Proposition 6.4.8	t_v	$O\left(m^{1/4}\sqrt{\frac{d_v}{t_v}}\right)$	$O(1)$
Bucket size S_i Proposition 6.4.10	$ B_i $	$O\left(\sqrt{\frac{n}{ B_i }}\right)$	$O^*\left(1 + \frac{m^{3/4}}{\sqrt{n\nu_i}}\right)$
Weighted triangle-degree \mathcal{T}_v Proposition 6.4.13	τ_v	$O\left(m^{1/4}\sqrt{\frac{d_v}{\tau_v}}\right)$	$O^*\left(1 + \frac{\sqrt{t_v}}{\sqrt{d_v}m^{1/4}} \frac{m^{3/4}}{\sqrt{t}}\right)$
Bucket weight W_i Proposition 6.4.15	$\sum_{v \in B_i} \tau_v$	$O\left(\sqrt{\frac{n}{ B_i }}\right)$	$O^*\left(1 + \frac{m^{3/4}}{\sqrt{n\nu_i}} + \sqrt{\frac{ B_i }{n}} \frac{m^{3/4}}{\sqrt{t}}\right)$

Table 6.1: Estimators used in the triangle counting algorithm. The values in the second column are approximately equal to the expectation of each estimator.

random variable $S_i = n\mathbb{1}_{v \in B_i}$ where v is a vertex chosen uniformly at random in the graph (Proposition 6.4.10). This approach poses two challenges. First, it requires a procedure to decide if a given vertex v belongs to the bucket B_i (which amounts to estimate if $t_v \in [\nu_i, \nu_{i+1})$). Secondly, the coefficient of variation of the random variable S_i is $\sqrt{n/|B_i|}$, which makes the time needed to estimate its mean prohibitively large when B_i is small. The first idea toward solving these problems is to restrict our attention to the *significant buckets* (Definition 6.4.3) defined as,

$$|B_i| \geq \Omega\left(\frac{\epsilon \bar{t}}{k\nu_i}\right) \quad \text{and} \quad \nu_i \leq O\left(\frac{\bar{t}^{2/3}}{\epsilon^{1/3}}\right). \quad (\text{significant})$$

We let \mathcal{S} denote the set of the vertices that belong to a significant bucket (the vertices in \mathcal{S} are also called *significant*). Although the non-significant vertices cannot be fully ignored (we explain later on how to compensate their loss), the sum $\frac{1}{3} \sum_{v \in \mathcal{S}} t_v$ of the triangle-degrees over \mathcal{S} already constitutes a constant fraction of the total triangle count t . This sum is broken down into different terms $\frac{1}{3} \sum_{v \in B_i} t_v$ for each significant bucket B_i . The latter quantity is now easier to estimate using the estimator $S_i = n\mathbb{1}_{v \in B_i}$. Indeed, the largeness condition on $|B_i|$ provides a lower bound on the coefficient of variation of S_i , and the smallness condition on the bucket boundary ν_i will facilitate the computation of whether a given vertex v belongs to B_i . We explain these two points in more detail in the next two paragraphs.

Triangle-degree estimator and bucket assignment. The triangle-degree t_v of a fixed vertex v can be represented as the expectation of a particular random variable T_v computed by a simple random process described in [ELRS17] and in Algorithm 6.4. The coefficient of variation of T_v is upper bounded by $O(m^{1/4}\sqrt{d_v/t_v})$, and the ℓ_2 -average stopping time of the process generating T_v is $O(1)$. We obtain a bucket assignment algorithm (Proposition 6.4.9) that decides if t_v lies in the interval $[\nu_i, \nu_{i+1})$ (meaning that v is assigned to the bucket B_i) by applying the *variable-time interval estimator* developed in the previous chapter to T_v . The query complexity of this algorithm is $O^*\left(1 + m^{1/4}\sqrt{d_v/\nu_i}\right)$.

Significant buckets detection. We use the above bucket assignment algorithm to implement the bucket size estimator $S_i = n\mathbb{1}_{v \in B_i}$ where v is chosen uniformly at random

(Proposition 6.4.10). The coefficient of variation of S_i is $O(\sqrt{n/|B_i|})$, and the ℓ_2 -average stopping time to compute S_i is $O^*\left(\sqrt{\frac{1}{n} \sum_{v \in V} (1 + m^{1/4} \sqrt{d_v/\nu_i})^2}\right) = O^*\left(1 + \frac{m^{3/4}}{\sqrt{n\nu_i}}\right)$. Consequently, by using again the **variable-time interval estimator**, we can decide if the size $|B_i|$ exceeds the threshold value $\Omega(\frac{\epsilon \bar{t}}{k\nu_i})$ in time $O^*\left(\sqrt{\frac{nk\nu_i}{\epsilon \bar{t}}} \left(1 + \frac{m^{3/4}}{\sqrt{n\nu_i}}\right)\right)$ (Proposition 6.4.11). We only perform this computation when $\nu_i \leq O\left(\frac{\bar{t}^{2/3}}{\epsilon^{1/3}}\right)$ to decide if the bucket is significant. Thus, the query complexity is $O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$. Furthermore, the size of a significant bucket can be estimated at the same cost by using the **variable-time exponential estimator**.

Weighted triangle-degree. The sum $\frac{1}{3} \sum_{v \in \mathcal{S}} t_v$ of the triangle-degrees over \mathcal{S} may be smaller than the lower endpoint $(1 - \epsilon)t$ of the error interval we are aiming at. Thus, we cannot simply estimate the size of the significant buckets for solving the problem (although it would be sufficient for getting a constant factor estimate of t). We address this issue by considering the set \mathcal{H} of all the *heavy* vertices (Definition 6.4.4), defined as

$$d_v \geq \Omega\left(\frac{\bar{m}}{\epsilon^{4/3} \bar{t}^{1/3}}\right) \quad \text{or} \quad t_v \geq \Omega\left(\frac{\bar{t}^{2/3}}{\epsilon^{1/3}}\right). \quad (\text{heavy})$$

We show that the augmented sum $\sum_{v \in \mathcal{S} \cup \mathcal{H}} t_v$ over the union of \mathcal{S} and \mathcal{H} is larger than the lower endpoint $(1 - \epsilon)t$ (Proposition 6.4.6). Thus, it can be estimated in place of the triangle count t . We estimate this sum in an indirect way by using a compensation idea. First, we assign the weight $w(\Delta) = 1/\max(1, 3 - h)$ to each triangle Δ in the graph, where $h \in \{0, 1, 2, 3\}$ is the number of heavy vertices contained in Δ (Definition 6.4.5). Next, we define the weighted triangle-degree τ_v of a vertex v as the sum of the weights of all the triangles adjacent to v . Suppose for a moment that none of the heavy vertices is significant, and that each triangle containing a heavy vertex also contains a significant one. Then the sum of the weighted triangle-degrees $\sum_{v \in \mathcal{S}} \tau_v$ over \mathcal{S} would exactly be equal to the sum of the triangle-degrees $\sum_{v \in \mathcal{S} \cup \mathcal{H}} t_v$ over $\mathcal{S} \cup \mathcal{H}$. In practice, a vertex can be both heavy and significant, and a triangle can contain three non-significant heavy vertices. However, we show that the total contribution of these events to the total triangle count is negligible (Proposition 6.4.6 and Proposition 6.4.7). Thus, the quantity $\sum_{v \in \mathcal{S}} \tau_v$ can be estimated in place of the triangle count t .

Weighted triangle-degree estimation. The weighted triangle-degree τ_v can be estimated in a similar way as the triangle-degree t_v (note that these two quantities can only differ by a factor of 3). The main difference is the use of a procedure for deciding whether some vertex w is heavy, which can be done in time $O^*(m^{3/4}/\sqrt{t})$ by using the **variable-time interval estimator** on the triangle-degree estimator T_w defined before (Proposition 6.4.12). The resulting algorithm (Proposition 6.4.13) generates a random variable \mathcal{T}_v whose expectation is close to τ_v , and whose coefficient of variation is upper bounded by a value $O(m^{1/4} \sqrt{d_v/\tau_v})$ similar to that of the triangle-degree estimator. The ℓ_2 -average stopping time of the quantum process generating \mathcal{T}_v is $O^*\left(1 + \frac{\sqrt{t_v}}{\sqrt{d_v} m^{1/4}} \frac{m^{3/4}}{\sqrt{t}}\right)$ due to the heavy detection procedure occurring with probability $\frac{t_v}{d_v \sqrt{m}}$ during the computation. As a result (Proposition 6.4.14), we can estimate with high accuracy the weighted triangle-degree τ_v of a vertex $v \in B_i$ in time $O^*\left(\frac{m^{1/4} \sqrt{d_v}}{\sqrt{\nu_i}} \left(1 + \frac{\sqrt{t_v}}{\sqrt{d_v} m^{1/4}} \frac{m^{3/4}}{\sqrt{t}}\right)\right) \leq O^*\left(\frac{m^{1/4} \sqrt{d_v}}{\sqrt{\nu_i}} + \frac{m^{3/4}}{\sqrt{t}}\right)$, by using the **variable-time relative estimator** on \mathcal{T}_v .

Bucket weight estimation. We can finally wrap up the algorithm. The triangle estimate \tilde{t} is set to be $\tilde{t} = \sum_i \tilde{w}_i$, where \tilde{w}_i is an estimate of the bucket weight $\sum_{v \in B_i} \tau_v$ for each significant bucket B_i (Proposition 6.4.17). The weight \tilde{w}_i is obtained by considering the unbiased bucket weight estimator $W_i = n \mathbb{1}_{v \in B_i} \tau_v$, where v is a vertex chosen uniformly at random in the graph (Proposition 6.4.15). We use the bucket assignment algorithm for computing $\mathbb{1}_{v \in B_i}$, and the weighted triangle-degree estimator for estimating τ_v when $v \in B_i$. The ℓ_2 -average stopping time of W_i is shown to be $O^*\left(1 + \frac{m^{3/4}}{\sqrt{nv_i}} + \sqrt{\frac{|B_i|}{n}} \frac{m^{3/4}}{\sqrt{t}}\right)$. The weight estimate \tilde{w}_i is obtained by using the **variable-time relative estimator** on W_i (Proposition 6.4.16). Since the coefficient of variation of W_i is upper bounded by $O(\sqrt{n/|B_i|})$, and provided that we only compute \tilde{w}_i for significant buckets, the query complexity is on the order of $O^*\left(\sqrt{\frac{n}{|B_i|}} \left(1 + \frac{m^{3/4}}{\sqrt{nv_i}} + \sqrt{\frac{|B_i|}{n}} \frac{m^{3/4}}{\sqrt{t}}\right)\right) \leq O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$.

Misclassification. A central aspect of the analysis, that we did not mention yet, is to handle the case where a vertex or a bucket is misclassified by the algorithm. This is likely to happen when the estimated quantity is close to the threshold value used to classify it. For instance, the bucket assignment procedure can wrongly assign a vertex v to a bucket B_i if the triangle-degree t_v is slightly smaller than the lower endpoint ν_i . We ensure that such misclassifications do not impact the accuracy of the triangle estimate much. First, the threshold values used to define the buckets and the sets \mathcal{S} and \mathcal{H} are randomly perturbed to guarantee that few elements lie in their neighborhood (Lemmas 6.4.1 and 6.4.2). Next, we duplicate each concept involving a threshold value (significant bucket, heavy vertex, triangle weight, etc.) into a *weak* variant allowing for a small error in the estimation. We then show that the algorithm remains correct in this relaxed setting.

Prior knowledge of \bar{m} and \bar{t} . We compute the edge estimate \bar{m} in time $O^*\left(\frac{\sqrt{n}}{m^{1/4}}\right)$ (Theorem 6.3.2) by using the **exponential estimator** on an unbiased estimator of m described in [Ses15] and in Algorithm 6.1. The triangle estimate $\bar{t} \in [t/8, t]$ is obtained in a more complicated way. We run our triangle counting algorithm over a decreasing sequence of values $\bar{t} = n^3, n^3/2, \dots$ (Algorithm 6.9). We show that the obtained estimate \tilde{t} is larger than $3\bar{t}$ when $\bar{t} > t$, and it becomes smaller than $3\bar{t}$ when $\bar{t} \in [t/8, t/4]$. This property is inherited from a Markov-like inequality satisfied by the quantum mean estimators, which states that the estimates are smaller than a small multiple of the mean with high probability. By comparing \bar{t} and \tilde{t} , we can detect when \bar{t} lies in the correct range of values.

Differences with [ELR15; Ses15; ELRS17]. Our work uses the quantum mean estimators developed in Chapter 5, whereas the classical algorithms use the median-of-means estimator. As a consequence, we must upper bound the ℓ_2 -average stopping time of the processes generating the random variables we are considering, whereas classically it suffices to bound the ℓ_1 -average stopping time. The bucketing technique originates from [GR08; GRS11; ELR15] but was abandoned in subsequent versions of the triangle counting algorithm [Ses15; ELRS17]. The weighted triangle-degree τ_v was introduced in the later version [Ses15; ELRS17], where the sum $\sum_{v \in \mathcal{S}} \tau_v$ is approximated by using a different estimator from W_i . The authors need to set up a data structure of size $O^*(n/t^{1/3})$ for sampling *edges* uniformly at random, which is unclear how to speed up in the quantum setting. In the present work, we combine the bucketing and the weighted triangle-degree ideas together to avoid edge sampling, which requires some subtle changes in the analysis. For instance, the definition of a heavy vertex differs by a factor of $1/\epsilon$ compared to [Ses15; ELRS17].

6.2 Preliminaries

Our algorithms are formulated in the graph query model. Before describing the latter, we first introduce the terminology used in the present chapter.

Definition 6.2.1 (TERMINOLOGY OF GRAPHS). A graph G with n vertices and m edges is a pair (V, E) , where $V = [n]$ is the vertex set and $E \subseteq \binom{V}{2}$ is the edge set of size m . We let \mathcal{G}_n denote the set of all graphs with n vertices. Given two vertices $v, w \in V$, we say that w is a *neighbor* of v if $\{v, w\} \in E$. An edge $e \in E$ is *adjacent* to a vertex v if $v \in e$. We let $E(v)$ denote the set of all the edges adjacent to v . The *degree* of a vertex v is $d_v = |E(v)|$. A *triangle* $\Delta = \{u, v, w\}$ is a subset of three vertices of V such that the edges $\{u, v\}$, $\{v, w\}$ and $\{u, w\}$ belong to E . We say that a triangle Δ is *adjacent* to v if $v \in \Delta$, and we let $T(v)$ denote the set of all the triangles adjacent to v . The *triangle-degree* t_v of v is defined as $t_v = |T(v)|$. The total number t of triangles in G is equal to $t = \frac{1}{3} \sum_{v \in V} t_v$.

We also define the following total order \prec on the vertex set $V = [n]$.

Definition 6.2.2 (VERTEX ORDERING). We let \prec denote the total order on V defined as $u \prec v$ when $d_u < d_v$, or when $d_u = d_v$ and $u < v$ (where $<$ is the natural order on $[n]$).

A graph can be represented in different ways, leading to different query models. The two most common representations are the *adjacency list* and *adjacency matrix* models, which are suitable for bounded degree graphs and dense graphs respectively. The *general graph model* [KKR04; Gol17] is a combination of these two models that is relevant when the input graph is arbitrary. We first present the classical definitions of these models.

Definition 6.2.3 (CLASSICAL GRAPH ORACLES). Given a graph $G = (V, E)$, we define three oracles to G corresponding to the following types of queries:

- (1) *degree query*: given $v \in V$, returns the degree d_v of v ,
- (2) *neighbor query*: given $v \in V$ and $i \in [n]$, returns the i -th neighbor of v (according to any fixed order) if $i \leq d_v$, and 0 otherwise,
- (3) *edge query*: given $u, v \in V$, returns the Boolean value indicating if $\{u, v\} \in E$.

A graph is in the *adjacency list* model if it can be accessed with degree and neighbor queries. It is in the *adjacency matrix* model if it can be accessed with edge queries. It is in the *general graph model* if it can be accessed with degree, neighbor and edge queries.

We adapt these models to the quantum setting by defining the corresponding quantum oracle operators. The query complexity of a quantum algorithm is equal to the number of times these operators are used.

Definition 6.2.4 (QUANTUM GRAPH ORACLES). Given a graph $G = (V, E)$, we consider the three unitary operators \mathcal{O}_{deg} , \mathcal{O}_{ngh} and \mathcal{O}_{edg} defined on the basis states as follows,

- (1) *degree query*: $\mathcal{O}_{\text{deg}}(|v\rangle|y\rangle) = |v\rangle|y \oplus d_v\rangle$, where $v \in V$ and $y \in \{0, 1\}^{\lceil \log n \rceil}$,
- (2) *neighbor query*: $\mathcal{O}_{\text{ngh}}(|v\rangle|i\rangle|y\rangle) = |v\rangle|i\rangle|y \oplus N(v, i)\rangle$, where $v \in V$, $i \in \{0, 1\}^{\lceil \log n \rceil}$, $y \in \{0, 1\}^{\lceil \log(n+1) \rceil}$ and $N(v, i)$ is the i -th neighbor of v if $i \leq d_v$, and 0 otherwise,
- (3) *edge query*: $\mathcal{O}_{\text{edg}}(|u\rangle|v\rangle|b\rangle) = |u\rangle|v\rangle|b \oplus \mathbb{1}_{\{u, v\} \in E}\rangle$ where $u, v \in V$ and $b \in \{0, 1\}$.

A graph is in the *quantum adjacency list* model if it can be accessed with \mathcal{O}_{deg} and \mathcal{O}_{ngh} . It is in the *quantum adjacency matrix* model if it can be accessed with \mathcal{O}_{edg} . It is in the *quantum general graph model* if it can be accessed with \mathcal{O}_{deg} , \mathcal{O}_{ngh} and \mathcal{O}_{edg} .

We recall that the notation $O^*(x)$ is used to hide any polynomial factor in $\log(x)$, $\log(n)$, $\log(1/\delta)$ and $1/\epsilon$, where δ is the failure probability parameter, and ϵ is the error parameter.

Quantum mean estimation. In this chapter, the *time complexity* (Assumption 5.A) is measured as the number of quantum queries to the input graph. We sometimes use the quantum mean estimators developed in Chapters 4 and 5 to estimate the expected output value of a *randomized* algorithm \mathcal{A} . When doing so, we implicitly assume that \mathcal{A} is first turned into a q-variable (Definition 4.2.3) by using standard reversibility techniques that do not significantly increase the time complexity (such as [Ben73]). In particular, if the number T of queries used by \mathcal{A} is a random variable, then we also call T the *stopping time* of \mathcal{A} and we can turn \mathcal{A} into a variable-time quantum algorithm with stopping time $O(T)$ (in the sense of Definition 5.2.2). Some of the quantum mean estimators also require the input q-random variable X to be distributed in $[0, 1]$. If X is instead distributed in $[0, M]$ for some $M \geq 1$, then we implicitly estimate the expectation of X/M and we multiply the result by M . This does not change the accuracy of the estimate since we use relative error bounds. Moreover, the time complexity increases only by a logarithmic factor in M .

6.3 Edge counting

We describe an optimal algorithm for estimating the number m of edges in the quantum adjacency list model. Our approach uses the following unbiased estimator of m taken from [Ses15]. The analysis is given here for completeness.

1. Sample a vertex $v \in V$ uniformly at random. Sample a neighbor w of v uniformly at random.
2. If $v \prec w$, then output nd_v . Else, output 0.

Algorithm 6.1: Edge estimator from [Ses15].

Proposition 6.3.1 (Theorem 12 in [Ses15]). *The output X of Algorithm 6.1 satisfies $\mathbb{E}[X] = m$ and $\text{Var}[X] \leq 2\sqrt{2}\sqrt{m}d_v t_v$. The query complexity is $O(1)$.*

Proof. Let d_v^+ denote the number of neighbors w of v such that $v \prec w$, where \prec is defined in Definition 6.2.2. We have $d_v^+ \leq \sqrt{2m}$ and $\sum_{v \in V} d_v^+ = m$. Thus, $\mathbb{E}[X] = \frac{1}{n} \sum_{v \in V} \frac{d_v^+}{d_v} nd_v = m$, and $\text{Var}[X] \leq \mathbb{E}[X^2] = \frac{1}{n} \sum_{v \in V} \frac{d_v^+}{d_v} (nd_v)^2 \leq n\sqrt{2m} \sum_{v \in V} d_v \leq 2\sqrt{2}nm^{3/2}$. \square

The classical edge counting algorithm [Ses15] consists of applying the *median-of-means* estimator to $O(\frac{n}{\epsilon^2\sqrt{m}})$ samples drawn from the random variable computed by Algorithm 6.1. We obtain a quadratic speedup by instead using the quantum exponential estimator.

Theorem 6.3.2 (EDGE COUNTING). *There exists a quantum algorithm with the following properties. Let G be a graph with n vertices and m edges in the quantum adjacency list model, and fix two parameters $\epsilon, \delta \in (0, 1/2)$. Then, the algorithm outputs an edge estimate \tilde{m} such that $|\tilde{m} - m| \leq \epsilon m$ with probability at least $1 - \delta$. The expected quantum query complexity of the algorithm is $O^*\left(\frac{\sqrt{n}}{m^{1/4}}\right)$.*

Proof. The algorithm consists of using the **exponential estimator** $\text{ExpEst}(X, f, \epsilon, \delta)$ on the random variable X computed by Algorithm 6.1 with $f : y \mapsto \frac{4\sqrt{n}}{y^{1/4}}$. Observe that $f(\mathbb{E}[X]) \geq \frac{\sqrt{\text{Var}[X]}}{\mathbb{E}[X]}$ by Proposition 6.3.1. Thus, the algorithm returns a correct estimate with probability at least $1 - \delta$ and uses $O^*\left(\frac{\sqrt{n}}{m^{1/4}}\right)$ queries in expectation by Corollary 4.5.6. \square

6.4 Triangle counting

In this section, we describe our quantum algorithm for estimating the number t of triangles in a graph. The relative error parameter $\epsilon \in (0, 1)$ is fixed from now on. We explain how to compute a triangle count estimate \tilde{t} such that $|\tilde{t} - t| \leq \epsilon t$ with probability at least $1 - 1/\log n$ (Theorem 6.4.18). We suppose that $t \geq 1$. A simple adaptation of our algorithm (that we will not describe here) can handle the case $t = 0$ separately by *finding* a triangle in time $\tilde{O}(\sqrt{n} + m^{3/4})$ if there is one.

6.4.1 Assumptions

We make four assumptions that simplify the description of the triangle counting algorithm. We explain later on how to remove them. First, we assume that the algorithm has prior knowledge of an edge estimate \bar{m} and a triangle estimate \bar{t} with constant relative error.

Assumption 6.A. The algorithm takes as input an edge estimate \bar{m} such that $\bar{m} \in [\frac{m}{4}, m]$.

Assumption 6.B. The algorithm takes as input a triangle estimate \bar{t} such that $\bar{t} \in [\frac{t}{8}, t]$.

Next, we assume that the algorithm has prior knowledge of a threshold value τ on the order of $\frac{t^{2/3}}{\epsilon^{1/3}}$ such that the sum of the triangle-degrees lying in a small neighborhood $[\tau^-, \tau^+]$ of τ is negligible. We explain how to satisfy this assumption in Lemma 6.4.1.

Assumption 6.C. The algorithm takes as input a threshold value $\tau > 0$ such that,

- (1) (*Separation*) Define $\tau^- = \left(1 - \frac{800\epsilon}{\log^3(n)}\right)\tau$ and $\tau^+ = \left(1 + \frac{800\epsilon}{\log^3(n)}\right)\tau$.
- (2) (*Triangle threshold*) $\tau^- \in \left[\frac{16\bar{t}^{2/3}}{\epsilon^{1/3}}, \frac{32\bar{t}^{2/3}}{\epsilon^{1/3}}\right]$.
- (3) (*Anti-concentration*) $\sum_{v: t_v \in [\tau^-, \tau^+]} t_v \leq \frac{\epsilon}{30}t$.

Finally, we consider two vertex bucketing sequences $(\hat{B}_i)_i$ and $(B_i)_i$, where each bucket contains all the vertices with a similar triangle-degree. We assume that $B_i \subseteq \hat{B}_i$ for all i , the sum of the triangle-degrees lying in $\hat{B}_i \setminus B_i$ is negligible, and there is some margin between the boundaries of B_i and \hat{B}_i . These buckets are represented in Figure 6.2. We explain how to construct them in Lemma 6.4.2.

Assumption 6.D. Let $k = \lfloor \frac{\log^4(n)}{90\epsilon} \rfloor$. The algorithm takes as input three increasing sequences $(\nu_i^-)_{i \in [k]}$, $(\nu_i^+)_{i \in [k]}$ and $(\nu_i)_{i \in [k]}$ where $\nu_i < \nu_i^- < \nu_i^+ < \nu_{i+1}$ for each i . Given the sets $B_i = \{v \in V : t_v \in [\nu_i^-, \nu_i^+]\}$ and $\hat{B}_i = \{v \in V : t_v \in [\nu_i, \nu_{i+1}]\}$, we have that

- (1) (*Partition & Inclusion*) $(\hat{B}_i)_i$ is a partition of $\{v \in V : t_v \neq 0\}$, and $B_i \subseteq \hat{B}_i$ for all i .
- (2) (*Closeness*) $\nu_{i+1} \leq \left(1 + \frac{800\epsilon}{\log^3(n)}\right)\nu_i$ for all i .
- (3) (*Separation*) $\nu_i \leq \left(1 - \frac{50\epsilon^3}{\log^8(n)}\right)\nu_i^-$ and $\nu_{i+1} \geq \left(1 + \frac{50\epsilon^3}{\log^8(n)}\right)\nu_i^+$ for all i .
- (4) (*Anti-concentration*) $\sum_{i \in [k]} \sum_{v \in \hat{B}_i \setminus B_i} t_v \leq \frac{\epsilon}{30}t$.

Assumptions 6.A and 6.B are removed in the final algorithm (Algorithm 6.9). Assumptions 6.C and 6.D are easier to satisfy. Indeed, it suffices to randomly choose the threshold values involved in their definition. We first describe a procedure that satisfies Assumption 6.C with high probability.

Lemma 6.4.1. *There exists an algorithm that outputs a threshold value $\tau > 0$ satisfying Assumption 6.C with probability at least $1 - \frac{1}{30 \log^2(n)}$. The algorithm makes no query.*

Proof. Define $\tau_j = (1 + \frac{1200\epsilon}{\log^3(n)})^{2j+1} \frac{16t^{2/3}}{\epsilon^{1/3}}$ and $\tau_j^- = (1 - \frac{800\epsilon}{\log^3(n)})\tau_j$, $\tau_j^+ = (1 + \frac{800\epsilon}{\log^3(n)})\tau_j$ for $j \geq 0$. Note that the intervals $[\tau_j^-, \tau_j^+]$ are disjoint. There are at most $\frac{90}{\epsilon}$ values j such that $\sum_{v: t_v \in [\tau_j^-, \tau_j^+]} t_v > \frac{\epsilon}{30}t$ since $\sum_{v \in V} t_v = 3t$. Consider the algorithm that chooses j uniformly at random between 0 and $\frac{2700 \log^2(n)}{\epsilon} - 1$ and that sets $\tau = \tau_j$, $\tau^- = \tau_j^-$, $\tau^+ = \tau_j^+$. By a union bound, this choice satisfies $\sum_{v: t_v \in [\tau^-, \tau^+]} t_v \leq \frac{\epsilon}{30}t$ with probability at least $1 - \frac{1}{30 \log^2(n)}$. Moreover, $\tau^- \leq (1 + \frac{1200\epsilon}{\log^3(n)})^{5400 \log^2(n)/\epsilon} \frac{16t^{2/3}}{\epsilon^{1/3}} \leq \frac{32t^{2/3}}{\epsilon^{1/3}}$ for n large enough. \square

We now describe a procedure that satisfies Assumption 6.D with high probability. The algorithm is similar to that of the previous lemma.

Lemma 6.4.2 (Adapted from [ELR15]). *There exists an algorithm that outputs three sequences $(\nu_i^-)_{i \in [k]}$, $(\nu_i^+)_{i \in [k]}$ and $(\nu_i)_{i \in [k]}$ satisfying Assumption 6.D with probability at least $1 - \frac{1}{2 \log n}$. The algorithm makes no query.*

Proof. For convenience in the proof, we change the range of i to $\{-1, 0, 1, \dots, k-2\}$. Let $\mu_i = (1 + \frac{800\epsilon}{\log^3(n)})^{i/2}$ for $i \geq -1$. We subdivide the interval $[\mu_i, \mu_{i+1}]$ by defining $\omega_{i,j} = \mu_i (1 + \frac{800\epsilon}{\log^3(n)})^{j\epsilon^2/(4 \log^5(n))}$ for $0 \leq j \leq 2 \log^5(n)/\epsilon^2$ (assuming $2 \log^5(n)/\epsilon^2$ is an integer). For each i , there are at most $\frac{90}{\epsilon}$ values j such that $\sum_{v: t_v \in [\omega_{i,j}, \omega_{i,j+1}]} t_v > \frac{\epsilon}{90} \sum_{v: t_v \in [\mu_i, \mu_{i+1}]} t_v$. Consider the algorithm that chooses j_i uniformly at random between 0 and $2 \log^5(n)/\epsilon^2 - 1$ for each i , and that sets $\nu_i^- = \omega_{i,j_i+1}$, $\nu_i^+ = \omega_{i+1,j_i+1}$ and $\nu_i = \omega_{i,j_i} (1 + \frac{800\epsilon}{\log^3(n)})^{\epsilon^2/(8 \log^5(n))}$. By a union bound, we have $\sum_{v: t_v \in [\omega_{i,j_i}, \omega_{i,j_i+1}]} t_v \leq \frac{\epsilon}{90} \sum_{v: t_v \in [\mu_i, \mu_{i+1}]} t_v$ for all i with probability at least $1 - \frac{90\epsilon}{2 \log^5(n)}k \geq 1 - \frac{1}{2 \log n}$. The construction is depicted in Figure 6.2.

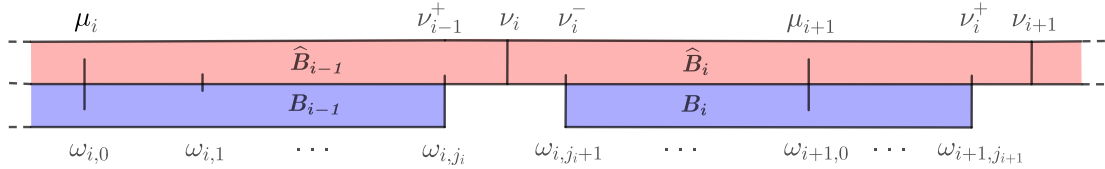


Figure 6.2: An illustration of the triangle-degree intervals defined in the proof of Lemma 6.4.2 (with a logarithmic scale). The upper red rectangles represent the buckets \hat{B}_i , whereas the lower blue ones represent the buckets B_i .

We prove that Assumption 6.D holds for the above choice with probability at least $1 - \frac{1}{2 \log n}$. First, for part (1), the buckets \hat{B}_i are disjoint and they cover all the vertices whose triangle-degree is between $\nu_{-1} \leq \mu_0 = 1$ and $\nu_{k-2} \geq \mu_{k-3} > n^2$. Thus, $(\hat{B}_i)_i$ is a partition of $\{v \in V : t_v \neq 0\}$. The inclusion property is trivial. Part (2) is a direct consequence of $\mu_i \leq \nu_i, \nu_{i+1} \leq \mu_{i+2} = (1 + \frac{800\epsilon}{\log^3(n)})\mu_i$. The first half of part (3) is obtained by $\nu_i = (1 + \frac{800\epsilon}{\log^3(n)})^{-\epsilon^2/(8 \log^5(n))} \nu_i^- \leq (1 - \frac{50\epsilon^3}{\log^8(n)})\nu_i$. The second half is obtained by $\nu_{i+1} = (1 + \frac{800\epsilon}{\log^3(n)})^{\epsilon^2/(8 \log^5(n))} \nu_i^+ \geq (1 + \frac{100\epsilon^3}{\log^8(n)})\nu_i^+$. Finally, for part (4), the sum of the triangle-degrees over the vertices outside a bucket B_i is at most $\sum_{i \in [k]} \sum_{v \in \hat{B}_i \setminus B_i} t_v = \sum_{i \in [k]} \sum_{v: t_v \in [\omega_{i,j_i}, \omega_{i,j_i+1}]} t_v \leq \frac{\epsilon}{90} \sum_{i \in [k]} \sum_{v: t_v \in [\mu_i, \mu_{i+1}]} t_v = \frac{\epsilon}{30}t$ with probability at least $1 - \frac{1}{2 \log n}$. \square

6.4.2 Main concepts

We introduce three concepts that play a central role in the triangle counting algorithm. First, we define the notion of “significant bucket” and “significant vertex”. A bucket is significant if it contains enough vertices, and if the triangle-degrees are sufficiently small. The notion of “weak significance” allows for a slightly smaller bucket size.

Definition 6.4.3 (SIGNIFICANT). A bucket B_i is *significant* if $|B_i| \geq \frac{\bar{\epsilon}t}{30k\nu_{i+1}}$ and $\nu_i \leq \tau$. A bucket \hat{B}_i is *weakly significant* if $|\hat{B}_i| \geq \frac{\bar{\epsilon}t}{60k\nu_{i+1}}$ and $\nu_i \leq \tau$. We let $I_S \subseteq [k]$ (resp. $I_{\hat{S}}$) denote the set of the indices of the significant (resp. weakly significant) buckets. A vertex v is (weakly) significant if it belongs to a (weakly) significant bucket. We let $\mathcal{S} = \cup_{i \in I_S} B_i$ denote the set of all significant vertices, and $\hat{\mathcal{S}} = \cup_{i \in I_{\hat{S}}} \hat{B}_i$ denote the set of all weakly significant vertices.

We say that a vertex is “heavy” if its degree or its triangle-degree is large. The notion of “weakly heavy” vertex allows for a slightly smaller triangle-degree.

Definition 6.4.4 (HEAVY). A vertex v is *heavy* if $d_v > \frac{2^{11}\bar{m}}{\epsilon^{4/3}\bar{t}^{1/3}}$ or $t_v > \tau$. A vertex v is *weakly heavy* if $d_v > \frac{2^{11}\bar{m}}{\epsilon^{4/3}\bar{t}^{1/3}}$ or $t_v > \tau^-$. We let \mathcal{H} (resp. $\hat{\mathcal{H}}$) denote the set of all heavy (resp. weakly heavy) vertices.

We finally define the “weight” of a triangle as the inverse of the number of non-heavy vertices it contains. We set the weight to 1 if all three vertices are heavy. The “weighted triangle-degree” of a vertex v is the sum of the weights of the triangles adjacent to v . We also adapt these definitions to the notion of “weakly heavy” vertices.

Definition 6.4.5 (WEIGHT). The *weight* $w(\Delta)$ of a triangle Δ is $w(\Delta) = 1/\max\{1, 3-h\}$, where h is the number of heavy vertices in Δ . The *weak weight* $\hat{w}(\Delta)$ of a triangle Δ is $\hat{w}(\Delta) = 1/\max\{1, 3-\hat{h}\}$, where \hat{h} is the number of weakly heavy vertices in Δ . Given a vertex v , we define the *weighted triangle-degree* $\tau_v = \sum_{\Delta \in T(v)} w(\Delta)$ and the *weakly weighted triangle-degree* $\hat{\tau}_v = \sum_{\Delta \in T(v)} \hat{w}(\Delta)$.

The triangle-degree t_v , the weighted triangle-degree τ_v and the weakly weighted triangle-degree $\hat{\tau}_v$ are related by the inequalities

$$\frac{1}{3}t_v \leq \tau_v \leq \hat{\tau}_v \leq t_v.$$

The total number t of triangles in the graph is equal to the sum of the triangle-degrees $t = \frac{1}{3} \sum_{v \in V} t_v$ over all the vertices. We show that the sum of the *weighted triangle-degrees* over the set of the *significant vertices* is equal to t up to an additive factor on the order of $\pm \epsilon t$ (a similar result is shown in [Ses15; ELRS17] for slightly different definitions). This is proved in two parts. First, we show that the latter sum is sufficiently large compared to t .

Proposition 6.4.6. *If Assumptions 6.A–6.D hold then $\sum_{v \in \mathcal{S}} \tau_v \geq (1 - \frac{4\epsilon}{5})t$.*

Proof. Given a vertex v such that $t_v \neq 0$, let $I(v) \in [k]$ be the index such that $v \in \hat{B}_{I(v)}$. The set $\mathcal{N} = V \setminus \mathcal{S}$ of all non-significant vertices is partitioned into the sets $\mathcal{N}_0 = V \setminus (\cup_{i \in [k]} B_i)$, $\mathcal{N}_1 = \left\{v : |B_{I(v)}| < \frac{\bar{\epsilon}t}{30k\nu_{2I(v)+1}}\right\} \setminus \mathcal{N}_0$ and $\mathcal{N}_2 = \left\{v : |B_{I(v)}| \geq \frac{\bar{\epsilon}t}{30k\nu_{2I(v)+1}} \text{ and } \nu_{2I(v)} > \tau\right\} \setminus \mathcal{N}_0$. The set \mathcal{H} of all heavy vertices is partitioned into the sets $\mathcal{H}_1 = \left\{v : d_v > \frac{2^{11}\bar{m}}{\epsilon^{4/3}\bar{t}^{1/3}} \text{ and } t_v \leq \tau\right\}$, $\mathcal{H}_2 = \{v \in \mathcal{N} : t_v > \tau\}$ and $\mathcal{H}_3 = \{v \in \mathcal{S} : t_v > \tau\}$. The relations between the different sets are depicted in Figure 6.3. We have $\mathcal{N}_2 \subseteq \mathcal{H}_2$.

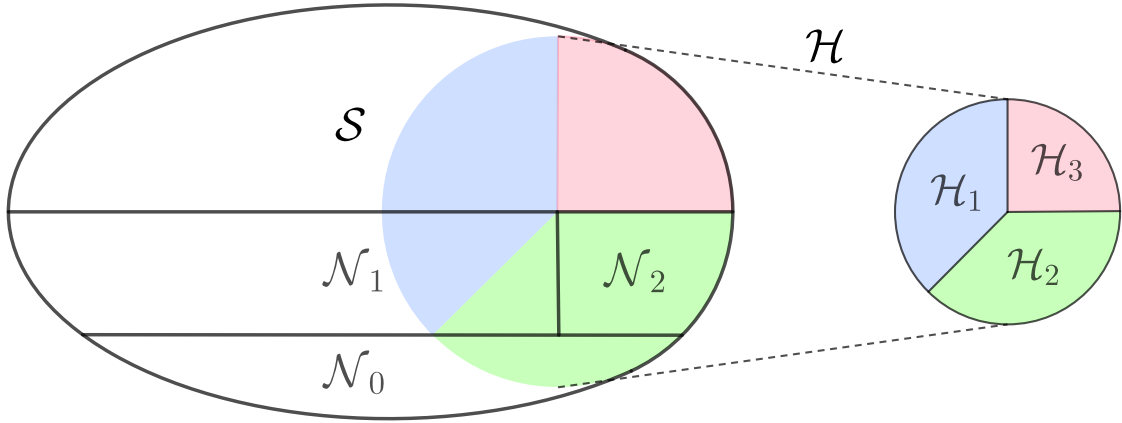


Figure 6.3: A Venn diagram of the sets described in the proof of Proposition 6.4.6. The ellipse represents the set V of all the vertices in the graph, where the upper half consists of the significant vertices (\mathcal{S}) and the lower half consists of the non-significant vertices ($\mathcal{N} = \mathcal{N}_0 \cup \mathcal{N}_1 \cup \mathcal{N}_2$). These sets are delimited by the black solid lines. The disk represents the set \mathcal{H} of all the heavy vertices, and it is partitioned into the three areas depicted on the right.

We prove that the sum of the triangle-degrees over $\mathcal{N}_0 \cup \mathcal{N}_1 \cup \mathcal{H}_1 \cup \mathcal{H}_3$ is at most $\frac{7}{20}\epsilon t$. For \mathcal{N}_0 , we have $\sum_{v \in \mathcal{N}_0} t_v \leq \frac{\epsilon}{30}t$ by parts (1) and (4) of Assumption 6.D. For \mathcal{N}_1 , we have $\sum_{v \in \mathcal{N}_1} t_v \leq \sum_{i \in [k]} \frac{\epsilon t}{30k\nu_{2I(v)+1}} \cdot \nu_{2I(v)+1} \leq \frac{\epsilon}{30}t$. The set \mathcal{H}_1 contains at most $\frac{2m\epsilon^{4/3}t^{1/3}}{2^{11\overline{m}}} \leq 2^{-8}\epsilon^{4/3}t^{1/3}\tau \leq 2^{-8}\epsilon^{4/3}t^{1/3}\frac{64t^{2/3}}{\epsilon^{1/3}} \leq \frac{\epsilon}{4}t$ by Assumption 6.C. The vertices in \mathcal{S} have triangle-degree at most $(1 + \frac{800\epsilon}{\log^3(n)})\tau = \tau^+$ by Definition 6.4.3, part (1) of Assumption 6.C, and part (2) of Assumption 6.D. Consequently, for all $v \in \mathcal{H}_3 \subset \mathcal{S}$, we have $t_v \in [\tau, \tau^+]$. Thus, by part (3) of Assumption 6.C, $\sum_{v \in \mathcal{H}_3} t_v \leq \frac{\epsilon}{30}t$. We conclude that $\sum_{v \in \mathcal{N}_0 \cup \mathcal{N}_1 \cup \mathcal{H}_1 \cup \mathcal{H}_3} t_v \leq (\frac{1}{30} + \frac{1}{30} + \frac{1}{4} + \frac{1}{30})\epsilon t = \frac{7}{20}\epsilon t$.

Let T_1 be the set of triangles that have all their vertices in \mathcal{H}_2 , let T_2 be the set of triangles that have at least one vertex in $\mathcal{N}_0 \cup \mathcal{N}_1 \cup \mathcal{H}_1 \cup \mathcal{H}_3$, and let T_3 be all the other triangles. The set \mathcal{H}_2 is of size at most $|\mathcal{H}_2| \leq \frac{3t}{\tau} \leq \frac{3}{4}(\epsilon t)^{1/3}$. Thus, $|T_1| \leq \binom{|\mathcal{H}_2|}{3} \leq (3/4)^3 \epsilon t$. We have shown in the previous paragraph that $|T_2| \leq \frac{7}{20}\epsilon t$. Finally, for any triangle $\Delta \in T_3$ there is an integer $h \in \{0, 1, 2\}$ such that Δ contains h vertices in $\mathcal{H}_2 \subseteq \mathcal{N}$ and $3-h$ vertices in $\mathcal{S} \setminus \mathcal{H}$, which implies that $\sum_{v \in \Delta \cap \mathcal{S}} w(\Delta) = 1$. Consequently, we have $\sum_{v \in \mathcal{S}} \tau_v \geq |T_3|$ and $t \leq |T_1| + |T_2| + |T_3| \leq \frac{4}{5}\epsilon t + |T_3|$. Thus, $\sum_{v \in \mathcal{S}} \tau_v \geq (1 - \frac{4\epsilon}{5})t$. \square

Next, we show that the above sum does not exceed the triangle count t by more than an additive factor on the order of ϵt . In fact, we prove the stronger result that the sum of the weakly weighted triangle-degrees over the larger set $\widehat{\mathcal{S}}$ of all weakly significant vertices is not too large. This result allows the triangle counting algorithm to misclassify a weakly significant or weakly heavy vertex as a significant or heavy vertex.

Proposition 6.4.7. *If Assumptions 6.A–6.D hold then $\sum_{v \in \widehat{\mathcal{S}}} \widehat{\tau}_v \leq (1 + \frac{3\epsilon}{5})t$.*

Proof. Given a vertex v such that $t_v \neq 0$, let $I(v) \in [k]$ be the index such that $v \in \widehat{B}_{I(v)}$. The set $\widehat{\mathcal{N}} = V \setminus \widehat{\mathcal{S}}$ of all non weakly significant vertices is partitioned into the sets $\widehat{\mathcal{N}}_0 = \{v \in V : t_v = 0\}$, $\widehat{\mathcal{N}}_1 = \left\{v : |\widehat{B}_{I(v)}| < \frac{\epsilon t}{60k\nu_{2I(v)+1}}\right\}$ and $\widehat{\mathcal{N}}_2 = \left\{v : |\widehat{B}_{I(v)}| \geq \frac{\epsilon t}{60k\nu_{2I(v)+1}} \text{ and } \nu_{2I(v)} > \tau\right\}$. The set $\widehat{\mathcal{H}}$ of all weakly heavy vertices is partitioned into

the sets $\hat{\mathcal{H}}_1 = \left\{v : d_v > \frac{2^{11}\bar{m}}{\epsilon^{4/3}\bar{t}^{1/3}} \text{ and } t_v \leq \tau^-\right\}$, $\hat{\mathcal{H}}_2 = \left\{v \in \hat{\mathcal{N}} : t_v > \tau^-\right\}$ and $\hat{\mathcal{H}}_3 = \left\{v \in \hat{\mathcal{S}} : t_v > \tau^-\right\}$. We have $\hat{\mathcal{N}}_2 \subseteq \hat{\mathcal{H}}_2$.

Similarly to the proof of Proposition 6.4.6, we have $\sum_{v \in \hat{\mathcal{N}}_1} t_v \leq \frac{\epsilon}{60}t$, $\sum_{v \in \hat{\mathcal{H}}_1} t_v \leq \frac{\epsilon}{8}t$ and $\sum_{v \in \hat{\mathcal{H}}_3} t_v \leq \frac{\epsilon}{30}t$. Let \hat{T}_1 be the set of triangles that have all their vertices in $\hat{\mathcal{H}}_2$, let \hat{T}_2 be the set of triangles that have at least one vertex in $\hat{\mathcal{N}}_1 \cup \hat{\mathcal{H}}_1 \cup \hat{\mathcal{H}}_3$, and let \hat{T}_3 be all the other triangles. For any $\Delta \in \hat{T}_1$, we have $\Delta \cap \hat{\mathcal{S}} = \emptyset$. Using the above inequalities, $|\hat{T}_2| \leq (\frac{1}{60} + \frac{1}{8} + \frac{1}{30})\epsilon t = \frac{7}{40}\epsilon t$. For any $\Delta \in \hat{T}_3$ there is an integer $\hat{h} \in \{0, 1, 2\}$ such that Δ contains \hat{h} vertices in $\hat{\mathcal{H}}_2 \subseteq \hat{\mathcal{N}}$ and $3 - \hat{h}$ vertices in $\hat{\mathcal{S}} \setminus \hat{\mathcal{H}}$, which implies that $\sum_{v \in \Delta \cap \hat{\mathcal{S}}} \hat{w}(\Delta) = 1$. Thus, $\sum_{v \in \hat{\mathcal{S}}} \hat{t}_v = \sum_{\Delta \in \hat{T}_1 \cup \hat{T}_2 \cup \hat{T}_3} \sum_{v \in \Delta \cap \hat{\mathcal{S}}} \hat{w}(\Delta) \leq 3|\hat{T}_2| + |\hat{T}_3| \leq (1 + \frac{3\epsilon}{5})t$. \square

6.4.3 Triangle degree estimator

The first ingredient of the algorithm is an unbiased estimator T_v of the triangle-degree t_v , taken from [ELRS17, Lemma 3.9]. It uses the order \prec on V described in Definition 6.2.2. The analysis of $\mathbb{E}[T_v]$ and $\text{Var}[T_v]$ was done in [ELRS17] (the proof is given here for completeness). The upper bound on the expected squared stopping time $\mathbb{E}[T^2]$ is new.

1. Sample an edge $e \in E(v)$ uniformly at random. Let w be the endpoint of e that is not v . Let u_e be the smaller endpoint of e according to \prec . Let $r = \lceil d_{u_e}/\sqrt{\bar{m}} \rceil$.
2. For $i = 1, \dots, r$:
 - a) Sample a neighbor x of u_e uniformly at random.
 - b) If e and x form a triangle and $w \prec x$, set $T_{v,i} = d_v d_{u_e}$. Else, set $T_{v,i} = 0$.
3. Output $T_v = \frac{1}{r} \sum_{i=1}^r T_{v,i}$.

Algorithm 6.4: Triangle-degree estimator, $\text{Triangle}(v)$.

Proposition 6.4.8. *Given a vertex v , if Assumption 6.A holds then the output T_v of $\text{Triangle}(v)$ (Algorithm 6.4) satisfies $\mathbb{E}[T_v] = t_v$ and $\text{Var}[T_v] \leq 3\sqrt{\bar{m}}d_v t_v$. The stopping time T of the algorithm satisfies $\mathbb{E}[T^2] \leq O(1)$.*

Proof. Given an edge $e = \{v, w\}$, let $t_{e,v}$ denote the number of triangles $\{v, w, x\}$ such that $w \prec x$. Observe that $t_v = \sum_{e \in E(v)} t_{e,v}$. Moreover, if $d_w \leq \sqrt{2\bar{m}}$ then $t_{e,v} \leq d_w \leq \sqrt{2\bar{m}}$, and if $d_w > \sqrt{2\bar{m}}$ then there are at most $\sqrt{2\bar{m}}$ neighbors x of w such that $w \prec x$. Thus, we always have $t_{e,v} \leq \sqrt{2\bar{m}}$.

The expectation of T_v is $\mathbb{E}[T_v] = \frac{1}{d_v} \sum_{e \in E(v)} \frac{t_{e,v}}{d_{u_e}} d_v d_{u_e} = t_v$. The variance of T_v conditioned on the edge e obtained at step 1 is $\text{Var}[T_v | e] = \frac{1}{\lceil d_{u_e}/\sqrt{\bar{m}} \rceil} \text{Var}[T_{v,1} | e] \leq \frac{1}{\lceil d_{u_e}/\sqrt{\bar{m}} \rceil} \mathbb{E}[T_{v,1}^2 | e] = \frac{1}{\lceil d_{u_e}/\sqrt{\bar{m}} \rceil} \frac{t_{e,v}}{d_{u_e}} (d_v d_{u_e})^2 \leq \sqrt{\bar{m}} d_v^2 t_{e,v}$. Thus, by the law of total variance, $\text{Var}[T_v] \leq \mathbb{E}[\text{Var}[T_v | e]] + \mathbb{E}[\mathbb{E}[T_v | e]^2] \leq \frac{1}{d_v} \sum_{e \in E(v)} \left(\sqrt{\bar{m}} d_v^2 t_{e,v} + \left(\frac{t_{e,v}}{d_{u_e}} d_v d_{u_e} \right)^2 \right) \leq (\sqrt{\bar{m}} + \sqrt{2\bar{m}}) d_v t_v \leq 3\sqrt{\bar{m}} d_v t_v$, where we used that $t_{e,v}^2 \leq \sqrt{2\bar{m}} t_{e,v}$. The number T of operations used by the algorithm satisfies $\mathbb{E}[T^2] \leq O\left(\frac{1}{d_v} \sum_{e \in E(v)} \left\lceil \frac{d_{u_e}}{\sqrt{\bar{m}}} \right\rceil^2\right) \leq O\left(1 + \frac{1}{d_v} \sum_{\{v,w\} \in E(v)} \frac{\min(d_v, d_w)^2}{\bar{m}}\right) \leq O\left(1 + \frac{1}{d_v} \sum_{\{v,w\} \in E(v)} \frac{d_v d_w}{\bar{m}}\right) \leq O(1)$. \square

We present three quantum algorithms that use the above estimator. First, we describe a bucket assignment algorithm that decides if a vertex v belongs to the bucket B_i . Due to the limited accuracy, the algorithm can output a wrong answer when $v \in \hat{B}_i \setminus B_i$.

Proposition 6.4.9. *There exists a quantum algorithm $\text{Bucket}(v, i, \delta)$ that takes as input a vertex v , an index $i \in [k]$ and a real $\delta \in (0, 1)$, and that outputs a Boolean value such that if Assumptions 6.A and 6.D hold then,*

- (1) *If $v \in B_i$ then the output is 1 with probability at least $1 - \delta$.*
- (2) *If $v \notin \hat{B}_i$ then the output is 0 with probability at least $1 - \delta$.*
- (3) *The query complexity of the algorithm is $O^*\left(1 + \frac{m^{1/4}\sqrt{d_v}}{\sqrt{\nu_i}}\right)$.*

Proof. Consider the function $f : y \mapsto \max(1, 5\bar{m}^{1/4}\sqrt{d_v/y})$. The algorithm consists of running the **variable-time interval estimator** $\text{VT-IntervEst}(T_v, f, T_2, \alpha, \beta, \epsilon', \delta)$ on the random variable T_v computed by $\text{Triangle}(v)$ (Proposition 6.4.8), where $\alpha = \nu_i^-$, $\beta = \nu_i^+$, $\epsilon' = \frac{50\epsilon^3}{\log^8(n)}$ and $T_2 = \Theta(1)$ is an upper bound on the ℓ_2 -average stopping time of $\text{Triangle}(v)$.

Observe that $f(\mathbb{E}[T_v]) \geq \max\left(1, \frac{\sqrt{\text{Var}[T_v]}}{\mathbb{E}[T_v]}\right)$. Thus, by Proposition 5.4.5, the output is 1 when $t_v \in [\nu_i^-, \nu_i^+]$ (i.e. $v \in B_i$) with probability at least $1 - \delta$, and it is 0 when $t_v \notin \left[\left(1 - \frac{50\epsilon^3}{\log^8(n)}\right)\nu_i^-, \left(1 + \frac{50\epsilon^3}{\log^8(n)}\right)\nu_i^+\right] \subset [\nu_i, \nu_{i+1}]$ (in particular, when $v \notin \hat{B}_i$) with probability at least $1 - \delta$. This proves parts (1) and (2) of the proposition. The query complexity is $O^*(f(\alpha)T_2) \leq O^*\left(1 + \frac{m^{1/4}\sqrt{d_v}}{\sqrt{\nu_i}}\right)$. \square

We use the previous bucket assignment algorithm to estimate the size $|B_i|$ of a given bucket B_i . The algorithm uses a quantum mean estimator on (an approximation of) the random variable $n\mathbb{1}_{v \in B_i}$ where v is a vertex chosen uniformly at random in the graph.

1. Sample a vertex $v \in V$ uniformly at random.
2. Run $\text{Bucket}(v, i, \delta)$ (Proposition 6.4.9), where $\delta = \frac{\epsilon^2}{2^{15}kn\nu_{i+1}}$, to decide if $v \in B_i$. If the result is 0 then output $S_i = 0$. Else, output $S_i = n$.

Algorithm 6.5: Bucket size estimator, $\text{BucketSize}(i)$.

Proposition 6.4.10. *Given an index $i \in [k]$, if Assumptions 6.A and 6.D hold then the output S_i of $\text{BucketSize}(i)$ (Algorithm 6.5) satisfies $\mathbb{E}[S_i] \in \left[\frac{3}{4}|B_i|, |\hat{B}_i| + \frac{\epsilon}{180k\nu_{i+1}}\right]$ and $\text{Var}[S_i] \leq n\mathbb{E}[S_i]$. The stopping time T of the algorithm satisfies $\mathbb{E}[T^2] \leq O^*\left(1 + \frac{m^{3/2}}{n\nu_i}\right)$.*

Proof. We only use $\delta \leq \frac{1}{4}$ and $\delta \leq \frac{\epsilon}{180k\nu_{i+1}}$ for that proof. If $v \in B_i$ then $\text{Bucket}(v, i, \delta)$ outputs 1 with probability at least $1 - \delta$ by Proposition 6.4.9. Thus, $\mathbb{E}[S_i] \geq \frac{1}{n} \sum_{v \in B_i} (1 - \delta)n \geq \frac{3}{4}|B_i|$. If $v \notin \hat{B}_i$ then $\text{Bucket}(v, i, \delta)$ outputs 0 with probability at least $1 - \delta$. Since $S_i \leq n$, we get $\mathbb{E}[S_i] \leq \delta n + \frac{1}{n} \sum_{v \in \hat{B}_i} n \leq \frac{\epsilon}{180k\nu_{i+1}} + |\hat{B}_i|$. The variance is $\text{Var}[S_i] \leq \mathbb{E}[S_i^2] \leq n\mathbb{E}[S_i]$. The stopping time satisfies $\mathbb{E}[T^2] \leq \frac{1}{n} \sum_{v \in V} O^*\left(1 + \frac{\sqrt{md_v}}{\nu_i}\right) \leq O^*\left(1 + \frac{m^{3/2}}{n\nu_i}\right)$. \square

Finally, we present an algorithm for deciding if a bucket is significant. The algorithm can output a wrong answer if the bucket is weakly significant but not significant. This is due to the inaccuracy of the bucket size estimation procedure.

Proposition 6.4.11. *There exists a quantum algorithm $\text{Significant}(i, \delta)$ that takes as input an index $i \in [k]$ and a real $\delta \in (0, 1)$, and that outputs a Boolean value such that if Assumptions 6.A–6.D hold then,*

- (1) *If $i \in I_S$ then the output is 1 with probability at least $1 - \delta$.*
- (2) *If $i \notin I_{\hat{S}}$ then the output is 0 with probability at least $1 - \delta$.*
- (3) *The query complexity of the algorithm is $O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$.*

Proof. We consider the algorithm that outputs 0 if $\nu_i > \tau$, and otherwise that outputs the result of the **variable-time interval estimator** $\text{VT-IntervEst}(S_i, f, T_2, \alpha, \beta, \epsilon', \delta)$ on the random variable S_i computed by $\text{BucketSize}(i)$ (Proposition 6.4.10), where $f : y \mapsto \sqrt{n/y}$, $\alpha = \frac{\epsilon t}{40k\nu_{i+1}}$, $\beta = +\infty$, $\epsilon' = \frac{4}{9}$ and $T_2 = O^*\left(1 + \frac{m^{3/4}}{\sqrt{n\nu_i}}\right)$ is an upper bound on the ℓ_2 -average stopping time of $\text{BucketSize}(i)$.

Note that $f(\mathbb{E}[S_i]) \geq \frac{\sqrt{\text{Var}[S_i]}}{\mathbb{E}[S_i]} \geq 1$ by Proposition 6.4.10. Moreover, if $i \in I_S$ then $\mathbb{E}[S_i] \geq \frac{3}{4}|B_i| \geq \frac{\epsilon t}{40k\nu_{i+1}} = \alpha$, and if $i \notin I_{\hat{S}}$ and $\nu_i \leq \tau$ then $\mathbb{E}[S_i] < |\hat{B}_i| + \frac{\epsilon}{180k\nu_{i+1}} \leq \frac{\epsilon t}{60k\nu_{i+1}} + \frac{\epsilon}{180k\nu_{i+1}} = \frac{\epsilon}{45k\nu_{i+1}} = \frac{5}{9}\alpha$. Thus, by Proposition 5.4.5, if $i \in I_S$ then the output is 1 with probability at least $1 - \delta$, and if $i \notin I_{\hat{S}}$ and $\nu_i \leq \tau$ then the output is 0 with probability at least $1 - \delta$. If $i \notin I_{\hat{S}}$ and $\nu_i > \tau$ then the output is 0 with probability 1. This proves parts (1) and (2) of the proposition. The query complexity is $O(1)$ when $\nu_i > \tau$, and $O^*(f(\alpha)T_2) \leq O^*\left(\sqrt{\frac{n\tau}{t}} + \frac{m^{3/4}}{\sqrt{t}}\right) \leq O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$ otherwise. \square

6.4.4 Weighted triangle degree estimator

We update the algorithms of the previous section to take the triangle weights into account. We first use the triangle estimator described before to decide if a vertex is heavy or not.

Proposition 6.4.12. *There exists a quantum algorithm $\text{Heavy}(v, \delta)$ that takes as input a vertex v and a real $\delta \in (0, 1)$, and that outputs a Boolean value such that if Assumptions 6.A–6.C hold then,*

- (1) *If $v \in \mathcal{H}$ then the output is 1 with probability at least $1 - \delta$.*
- (2) *If $v \notin \hat{\mathcal{H}}$ then the output is 0 with probability at least $1 - \delta$.*
- (3) *The query complexity of the algorithm is $O^*\left(\frac{m^{3/4}}{\sqrt{t}}\right)$.*

Proof. The proof is similar to that of Proposition 6.4.9. Consider the function $f : y \mapsto \max\left(1, \frac{2^7 \bar{m}^{3/4}}{\epsilon^{2/3} \bar{t}^{1/6} \sqrt{y}}\right)$. The algorithm consists of two steps. First, it queries the degree of v and it outputs 1 if $d_v > \frac{2^{11} \bar{m}}{\epsilon^{4/3} \bar{t}^{1/3}}$. Else, it outputs the result of the **variable-time interval estimator** $\text{VT-IntervEst}(T_v, f, T_2, \alpha, \beta, \epsilon', \delta)$ on the random variable T_v computed by $\text{Triangle}(v)$ (Proposition 6.4.8), where $\alpha = \tau$, $\beta = +\infty$, $\epsilon' = \frac{800\epsilon}{\log^3(n)}$ and $T_2 = \Theta(1)$ is an upper bound on the ℓ_2 -average stopping time of $\text{Triangle}(v)$.

The algorithm is always correct when $d_v > \frac{2^{11} \bar{m}}{\epsilon^{4/3} \bar{t}^{1/3}}$. If $d_v \leq \frac{2^{11} \bar{m}}{\epsilon^{4/3} \bar{t}^{1/3}}$ then $\frac{\sqrt{\text{Var}[T_v]}}{\mathbb{E}[T_v]} \leq \sqrt{3}m^{1/4} \sqrt{\frac{d_v}{t_v}} \leq \frac{2^7 \bar{m}^{3/4}}{\epsilon^{2/3} \bar{t}^{1/6} \sqrt{t_v}}$ by Proposition 6.4.8. Thus, the function f satisfies $f(\mathbb{E}[T_v]) \geq \max\left(1, \frac{\sqrt{\text{Var}[T_v]}}{\mathbb{E}[T_v]}\right)$. By Proposition 5.4.5, if $t_v \in [\tau, +\infty)$ (i.e. $v \in \mathcal{H}$) the output is 1 with probability at least $1 - \delta$, and if $t_v \leq \tau^- = \left(1 - \frac{800\epsilon}{\log^3(n)}\right)\tau$ (i.e. $v \notin \hat{\mathcal{H}}$) the output is 0 with probability at least $1 - \delta$. The query complexity is $O^*(f(\alpha)T_2) \leq O^*(m^{3/4}/\sqrt{t})$. \square

We use the above algorithm to construct an estimator for the weighted triangle-degree τ_v . This is a refinement of the triangle estimator, adapted from [ELRS17, Claim 3.16].

1. Sample an edge $e \in E(v)$ uniformly at random. Let w be the endpoint of e that is not v . Let u_e be the smaller endpoint of e according to \prec .
2. If $d_{u_e} \leq \sqrt{m}$, set $r = 1$ with probability d_{u_e}/\sqrt{m} , and output $\mathcal{T}_v = 0$ otherwise. If $d_{u_e} > \sqrt{m}$, set $r = \lceil d_{u_e}/\sqrt{m} \rceil$.
3. For $i = 1, \dots, r$:
 - a) Sample a neighbor x of u_e uniformly at random.
 - b) If e and x do not form a triangle, or if $x \prec w$, set $\mathcal{T}_{v,i} = 0$.
 - c) Else, compute $h = \text{Heavy}(v, \frac{\epsilon}{180}) + \text{Heavy}(w, \frac{\epsilon}{180}) + \text{Heavy}(x, \frac{\epsilon}{180})$, and set $\mathcal{T}_{v,i} = \frac{d_v \max\{d_{u_e}, \sqrt{m}\}}{\max\{1, 3-h\}}$.
4. Output $\mathcal{T}_v = \frac{1}{r} \sum_{i=1}^r \mathcal{T}_{v,i}$.

Algorithm 6.6: Weighted triangle-degree estimator, $\text{WeightedTriangle}(v)$.

Proposition 6.4.13. *Given a vertex v , if Assumptions 6.A–6.C hold then the output \mathcal{T}_v of $\text{WeightedTriangle}(v)$ (Algorithm 6.6) satisfies $\mathbb{E}[\mathcal{T}_v] \in [(1 - \frac{\epsilon}{20})\tau_v, (1 + \frac{\epsilon}{20})\hat{\tau}_v]$ and $\text{Var}[\mathcal{T}_v] \leq 7\sqrt{m}d_v t_v$. The stopping time T of the algorithm satisfies $\mathbb{E}[T^2] \leq O^*\left(1 + \frac{t_v}{d_v\sqrt{m}} \frac{m^{3/2}}{t}\right)$.*

Proof. The proof is similar to that of Proposition 6.4.8. For any triangle $\Delta = \{v, w, x\}$, let $\hat{w}(\Delta) = \frac{1}{\max\{1, 3-h\}}$ denote the approximate triangle weight obtained with the value h computed at step 3.c. We have $\mathbb{E}[\hat{w}(\Delta)] \geq (1 - \frac{\epsilon}{180})^3 w(\Delta) \geq (1 - \frac{\epsilon}{60})w(\Delta)$ and $\mathbb{E}[\hat{w}(\Delta)] \leq \hat{w}(\Delta) + \frac{3\epsilon}{180} \leq (1 + \frac{\epsilon}{20})\hat{w}(\Delta)$ by property of **Heavy** (Proposition 6.4.12) and the fact that $\frac{1}{3} \leq w(\Delta) \leq \hat{w}(\Delta) \leq 1$. Given an edge $e = \{v, w\}$, let $T(e, v)$ denote the set of all triangles that contribute to $t_{e,v}$ (where $t_{e,v}$ is defined in the proof of Proposition 6.4.8). The expectation of \mathcal{T}_v conditioned on the edge e obtained at step 1 and $d_{u_e} \leq \sqrt{m}$ satisfies $\mathbb{E}[\mathcal{T}_v | e, d_{u_e} \leq \sqrt{m}] \leq \frac{d_{u_e}}{\sqrt{m}} \cdot \frac{1}{d_{u_e}} \sum_{\Delta \in T(e,v)} d_v \sqrt{m} (1 + \frac{\epsilon}{20}) \hat{w}(\Delta) = (1 + \frac{\epsilon}{20}) d_v \sum_{\Delta \in T(e,v)} \hat{w}(\Delta)$. The expectation of \mathcal{T}_v conditioned on the edge e obtained at step 1 and $d_{u_e} > \sqrt{m}$ satisfies $\mathbb{E}[\mathcal{T}_v | e, d_{u_e} > \sqrt{m}] \leq \frac{1}{d_{u_e}} \sum_{\Delta \in T(e,v)} d_v d_{u_e} (1 + \frac{\epsilon}{20}) \hat{w}(\Delta) = (1 + \frac{\epsilon}{20}) d_v \sum_{\Delta \in T(e,v)} \hat{w}(\Delta)$. Thus, by the law of total expectation, $\mathbb{E}[\mathcal{T}_v] \leq \frac{1}{d_v} \sum_{e \in E(v)} (1 + \frac{\epsilon}{20}) d_v \sum_{\Delta \in T(e,v)} \hat{w}(\Delta) = (1 + \frac{\epsilon}{20}) \hat{\tau}_v$. The lower bound $\mathbb{E}[\mathcal{T}_v] \geq (1 - \frac{\epsilon}{20})\tau_v$ is proved in a similar way. The variance satisfies $\text{Var}[\mathcal{T}_v | e, d_{u_e} \leq \sqrt{m}] \leq \mathbb{E}[\mathcal{T}_v^2 | e, d_{u_e} \leq \sqrt{m}] \leq \frac{d_{u_e}}{\sqrt{m}} \cdot \frac{t_{e,v}}{d_{u_e}} \cdot (d_v \sqrt{m})^2 = \sqrt{m} d_v^2 t_{e,v}$ and $\text{Var}[\mathcal{T}_v | e, d_{u_e} > \sqrt{m}] \leq \frac{\sqrt{m}}{d_{u_e}} \mathbb{E}[\mathcal{T}_{v,1}^2 | e] \leq \frac{\sqrt{m}}{d_{u_e}} \cdot \frac{t_{e,v}}{d_{u_e}} \cdot (d_v d_{u_e})^2 \leq \sqrt{m} d_v^2 t_{e,v}$. Thus, by the law of total variance, $\text{Var}[\mathcal{T}_v] \leq \mathbb{E}[\text{Var}[\mathcal{T}_v | e]] + \mathbb{E}[\mathbb{E}[\mathcal{T}_v | e]^2] \leq \frac{1}{d_v} \sum_{e \in E(v)} \left(\sqrt{m} d_v^2 t_{e,v} + ((1 + \frac{\epsilon}{20}) d_v t_{e,v})^2 \right) \leq 7\sqrt{m} d_v \sum_{e \in E(v)} t_{e,v} = \sqrt{m} d_v t_v$, where we used that $t_{e,v}^2 \leq \sqrt{2m} t_{e,v}$.

The expectation of T^2 conditioned on the edge e sampled at step 1 and $d_{u_e} \leq \sqrt{m}$ is $\mathbb{E}[T^2 | e, d_{u_e} \leq \sqrt{m}] \leq O(1 + \frac{d_{u_e}}{\sqrt{m}} \cdot \frac{t_{e,v}}{d_{u_e}} T_{\text{Heavy}}^2) = O(1 + \frac{t_{e,v}}{\sqrt{m}} T_{\text{Heavy}}^2)$ where $T_{\text{Heavy}} \leq O^*\left(\frac{m^{3/4}}{\sqrt{t}}\right)$ is the query complexity of **Heavy**. If we condition instead on $d_{u_e} > \sqrt{m}$ then the number of executions of step 3.c is upper bounded by the outcome of the binomial distribution $B\left(\left\lceil \frac{d_{u_e}}{\sqrt{m}} \right\rceil, \frac{t_{e,v}}{d_{u_e}}\right)$, whose second moment is at most $\left(\left\lceil \frac{d_{u_e}}{\sqrt{m}} \right\rceil \cdot \frac{t_{e,v}}{d_{u_e}}\right)^2 + \left\lceil \frac{d_{u_e}}{\sqrt{m}} \right\rceil \cdot \frac{t_{e,v}}{d_{u_e}} \leq O\left(\frac{t_{e,v}}{\sqrt{m}}\right)$.

since $d_{u_e} \geq \sqrt{m}$ and $t_{e,v} \leq \sqrt{2m}$. Thus, $\mathbb{E}[T^2 \mid e, d_{u_e} > \sqrt{m}] \leq O(1 + \frac{t_{e,v}}{\sqrt{m}} T_{\text{Heavy}}^2)$. By the law of total expectation, $\mathbb{E}[T^2] \leq \frac{1}{d_v} \sum_{e \in E(v)} O(1 + \frac{t_{e,v}}{\sqrt{m}} T_{\text{Heavy}}^2) \leq O(1 + \frac{t_v}{d_v \sqrt{m}} T_{\text{Heavy}}^2)$. \square

We apply a quantum mean estimator to the above algorithm in order to estimate τ_v with high accuracy. We suppose that the index $i \in [k]$ for which $v \in \hat{B}_i$ is known.

Proposition 6.4.14. *There is a quantum algorithm `ApproxWeightedTriangle`(v, i, δ) that takes as input an index $i \in [k]$, a vertex $v \in \hat{B}_i$ and a real $\delta \in (0, 1)$, and that outputs an estimate $\tilde{\tau}_v$ such that if Assumptions 6.A–6.D hold then $\tilde{\tau}_v \in [(1 - \frac{\epsilon}{15})\tau_v, (1 + \frac{\epsilon}{15})\hat{\tau}_v]$ with probability at least $1 - \delta$. The query complexity of the algorithm is $O^*\left(\frac{\sqrt{d_v} m^{1/4}}{\sqrt{\nu_i}} + \frac{m^{3/4}}{\sqrt{t}}\right)$.*

Proof. The estimate $\tilde{\tau}_v$ is obtained by running the **variable-time relative estimator** `VT-RelatEst`($\mathcal{T}_v, \mathcal{V}, T_2, \alpha, \frac{\epsilon}{70}, \delta$) on the random variable \mathcal{T}_v output by `WeightedTriangle`(v) (Proposition 6.4.13) with $\mathcal{V} = \max\left(1, \frac{10\sqrt{d_v} m^{1/4}}{\sqrt{\nu_i}}\right)$, $\alpha = \nu_i/6$ and $T_2 = O^*\left(1 + \frac{\sqrt{\nu_i}}{\sqrt{d_v} m^{1/4}} \frac{m^{3/4}}{\sqrt{t}}\right)$ being an upper bound on the ℓ_2 -average stopping time of `WeightedTriangle`(v).

Note that $\mathcal{V} \geq \max\left(1, \sqrt{\text{Var}[\mathcal{T}_v]/\mathbb{E}[\mathcal{T}_v]}\right)$ and $\mathbb{E}[\mathcal{T}_v] \geq (1 - \frac{\epsilon}{20})\tau_v \geq t_v/6 \geq \alpha$ since $v \in \hat{B}_i$ and by Proposition 6.4.13. Thus, the estimate $\tilde{\tau}_v$ satisfies $|\tilde{\tau}_v - \mathbb{E}[\mathcal{T}_v]| \leq \frac{\epsilon}{70} \mathbb{E}[\mathcal{T}_v]$ with probability at least $1 - \delta$. Since $\mathbb{E}[\mathcal{T}_v] \in [(1 - \frac{\epsilon}{20})\tau_v, (1 + \frac{\epsilon}{20})\hat{\tau}_v]$, we obtain that $\tilde{\tau}_v \in [(1 - \frac{\epsilon}{20})(1 - \frac{\epsilon}{70})\tau_v, (1 + \frac{\epsilon}{20})(1 + \frac{\epsilon}{70})\hat{\tau}_v] \subset [(1 - \frac{\epsilon}{15})\tau_v, (1 + \frac{\epsilon}{15})\hat{\tau}_v]$ with probability at least $1 - \delta$. The query complexity is $O^*(\mathcal{V}T_2) \leq O^*\left(\frac{\sqrt{d_v} m^{1/4}}{\sqrt{\nu_i}} + \frac{m^{3/4}}{\sqrt{t}}\right)$. \square

Similarly to the bucket size estimator of Proposition 6.4.10, we construct a bucket weight estimator for the quantity $\sum_{v \in B_i} \tau_v$ by using the previous proposition. We restrict our attention to the case of weakly significant buckets. The new estimator differs from Algorithm 6.5 by multiplying the final result with a triangle-weight estimate.

1. Sample a vertex $v \in V$ uniformly at random.
2. Run `Bucket`(v, i, δ) (Proposition 6.4.9), where $\delta = \frac{\epsilon^2}{2^{15} k n \nu_{i+1}}$, to decide if $v \in \hat{B}_i$,
 - a) If the result is 0 then output $W_i = 0$.
 - b) Else, compute a weight estimate $\tilde{\tau}_v$ by using `ApproxWeightedTriangle`(v, i, δ) (Proposition 6.4.14). Output $W_i = n \cdot \text{median}\{\nu_i/3, \tilde{\tau}_v, \nu_{i+1}\}$.

Algorithm 6.7: Bucket weight estimator, `BucketWeight`(i).

Proposition 6.4.15. *Given an index $i \in I_{\hat{S}}$, let W_i denote the output of `BucketWeight`(i) (Algorithm 6.7) and let S_i denote the output of `BucketSize`(i) (Proposition 6.4.10). Then,*

- (1) *If Assumptions 6.A–6.D hold then $\mathbb{E}[W_i] \in \left[(1 - \frac{\epsilon}{10}) \sum_{v \in B_i} \tau_v, (1 + \frac{\epsilon}{10}) \sum_{v \in \hat{B}_i} \hat{\tau}_v\right]$ and $\text{Var}[W_i] \leq n \nu_{i+1} \mathbb{E}[W_i]$.*
- (2) *If Assumptions 6.A and 6.D hold then $\mathbb{E}[W_i] \in \left[\frac{\nu_i}{3} \mathbb{E}[S_i], \nu_{i+1} \mathbb{E}[S_i]\right]$.*
- (3) *If Assumptions 6.A–6.D hold then the stopping time T of `BucketWeight`(i) satisfies $\mathbb{E}[T^2] \leq O^*\left(1 + \frac{m^{3/2}}{n \nu_i} + \frac{\mathbb{E}[S_i]}{n} \frac{m^{3/2}}{t}\right)$.*

Proof. The algorithm $\text{BucketWeight}(i)$ is identical to $\text{BucketSize}(i)$ (Proposition 6.4.10), except at step 2.b where the output is multiplied by a coefficient in the interval $[\nu_i/3, \nu_{i+1}]$. This directly proves part (2) of the proposition. The variance is $\text{Var}[W_i] \leq \mathbb{E}[W_i^2] \leq n\nu_{i+1}\mathbb{E}[W_i]$ since $W_i \leq n\nu_{i+1}$. For the rest of the proof, observe that for any vertex $v \in \hat{B}_i$ the weighted triangle-degrees τ_v and $\hat{\tau}_v$ must lie in the interval $[\nu_i/3, \nu_{i+1}]$ since $t_v \in [\nu_i, \nu_{i+1}]$ and $\tau_v, \hat{\tau}_v \in [t_v/3, t_v]$.

We first prove the lower bound $\mathbb{E}[W_i] \geq (1 - \frac{\epsilon}{10}) \sum_{v \in B_i} \tau_v$ given in part (1). If $v \in B_i$ then the algorithm $\text{Bucket}(v, i, \delta)$ outputs 1 with probability at least $1 - \delta$ (Proposition 6.4.9) and the estimate $\tilde{\tau}_v$ satisfies $\tilde{\tau}_v \geq (1 - \frac{\epsilon}{15})\tau_v$ with probability at least $1 - \delta$ (Proposition 6.4.14). Thus, $\mathbb{E}[W_i] \geq (1 - \delta)^2 \frac{1}{n} \sum_{v \in B_i} n(1 - \frac{\epsilon}{15})\tau_v \geq (1 - \frac{\epsilon}{10}) \sum_{v \in B_i} \tau_v$ by our choice of δ .

Next, we prove the upper bound $\mathbb{E}[W_i] \leq (1 + \frac{\epsilon}{10}) \sum_{v \in \hat{B}_i} \hat{\tau}_v$ given in part (1). If $v \notin \hat{B}_i$ then $\text{Bucket}(v, i, \delta)$ outputs 0 with probability at least $1 - \delta$ (Proposition 6.4.9). If $v \in \hat{B}_i$ then the estimate $\tilde{\tau}_v$ satisfies $\tilde{\tau}_v \leq (1 + \frac{\epsilon}{15})\hat{\tau}_v$ with probability at least $1 - \delta$ (Proposition 6.4.14). Since $W_i \leq n\nu_{i+1}$, we have $\mathbb{E}[W_i] \leq \delta n\nu_{i+1} + \frac{1}{n} \sum_{v \in \hat{B}_i} n(1 + \frac{\epsilon}{15})\hat{\tau}_v$. Moreover, $\sum_{v \in \hat{B}_i} \hat{\tau}_v \geq |\hat{B}_i| \frac{\nu_i}{3} \geq \frac{\epsilon t}{360k} \geq \frac{30}{\epsilon} \cdot \delta n\nu_{i+1}$ since $i \in I_{\hat{S}}$. Thus, by our choice of δ , we have $\mathbb{E}[W_i] \leq \frac{\epsilon}{30} \sum_{v \in \hat{B}_i} \hat{\tau}_v + (1 + \frac{\epsilon}{15}) \sum_{v \in \hat{B}_i} \hat{\tau}_v \leq (1 + \frac{\epsilon}{10}) \sum_{v \in \hat{B}_i} \hat{\tau}_v$.

The stopping time of $\text{BucketWeight}(i)$ can increase compared to that of $\text{BucketSize}(i)$ only at step 2.b. The probability to run this step is equal to $\frac{\mathbb{E}[S_i]}{n}$ by definition of Algorithm 6.5 and Algorithm 6.7. Moreover, the query complexity of $\text{ApproxWeightedTriangle}(v, i, \delta)$ is $O^*\left(\frac{\sqrt{d_v}m^{1/4}}{\sqrt{\nu_i}} + \frac{m^{3/4}}{\sqrt{t}}\right)$ by Proposition 6.4.14. The first term in this expression is of the same order of magnitude as the complexity of $\text{Bucket}(v, i, \delta)$ (used at step 2), and thus can be ignored. By adding the contribution of the second term to the complexity of $\text{BucketSize}(i)$, we obtain that $\mathbb{E}[T^2] \leq O^*\left(1 + \frac{m^{3/2}}{n\nu_i} + \frac{\mathbb{E}[S_i]}{n} \frac{m^{3/2}}{t}\right)$. \square

Finally, we describe the main algorithm for estimating the total weight $\sum_{v \in B_i} \tau_v$ of a significant bucket B_i with any desired accuracy (Algorithm 6.8). The algorithm first ensures that the bucket is significant (step 1) before estimating its weight (step 3). We apply a quantum mean estimator (step 3.b) on the bucket weight estimator from Proposition 6.4.15. The upper bounds on the coefficient of variation and on the ℓ_2 -average stopping time of the latter estimator are obtained at step 3.a by computing a constant factor estimate \tilde{s}_i of the bucket size $|B_i|$.

Proposition 6.4.16. *Given an index $i \in [k]$ and a real $\delta \in (0, 1)$, the bucket weight estimate \tilde{w}_i output by $\text{ApproxBucketWeight}(i, \delta)$ (Algorithm 6.8) satisfies the following properties. If Assumptions 6.A–6.D hold then,*

- (1) *If $i \in I_S$ then $\tilde{w}_i \geq (1 - \frac{3\epsilon}{20}) \sum_{v \in B_i} \tau_v$ with probability at least $1 - \delta$.*
- (2) *If $i \in I_{\hat{S}}$ then $\tilde{w}_i \leq (1 + \frac{3\epsilon}{20}) \sum_{v \in \hat{B}_i} \hat{\tau}_v$ with probability at least $1 - \delta$.*
- (3) *If $i \notin I_{\hat{S}}$ then $\tilde{w}_i = 0$ with probability at least $1 - \delta$.*
- (4) *The query complexity of the algorithm is $O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$ with probability at least $1 - \delta$ when $i \in I_S$ or $i \notin I_{\hat{S}}$.*

If Assumptions 6.A and 6.D hold then,

- (5) *$\tilde{w}_i \leq 2\nu_{i+1}|\hat{B}_i| + \frac{\epsilon}{90k}$ with probability at least $1 - \delta$.*

1. Run $\text{Significant}(i, \delta/3)$ (Proposition 6.4.11) to decide if i is significant.
2. If the outcome of step 1 is 0 then output $\tilde{w}_i = 0$.
3. Else,
 - a) Compute a bucket size estimate \tilde{s}_i by using the **variable-time exponential estimator** $\text{VT-ExpEst}(S_i, f, T_2, \frac{1}{2}, \frac{\delta}{3})$ on the random variable S_i output by $\text{BucketSize}(i)$ (Proposition 6.4.10), where $f : y \mapsto \sqrt{n/y}$, and $T_2 = O^*\left(\sqrt{1 + \frac{\overline{m}^{3/2}}{n\nu_i}}\right)$.
 - b) Compute a bucket weight estimate \tilde{w}_i by running the **variable-time relative estimator** $\text{VT-RelatEst}(W_i, \mathcal{V}, T_2, \alpha, \frac{\epsilon}{25}, \frac{\delta}{3})$ on the random variable W_i output by $\text{BucketWeight}(i)$ (Proposition 6.4.15), where $\mathcal{V} = \sqrt{5n/\tilde{s}_i}$, $\alpha = \nu_i \tilde{s}_i/5$ and $T_2 = O^*\left(\sqrt{1 + \frac{\overline{m}^{3/2}}{n\nu_i} + \frac{2\tilde{s}_i}{n} \frac{\overline{m}^{3/2}}{t}}\right)$.
 - c) Output \tilde{w}_i .

Algorithm 6.8: Bucket weight estimation, $\text{ApproxBucketWeight}(i, \delta)$.

Proof. If $i \notin I_{\mathcal{S}}$ then the algorithm stops at step 2 and outputs $\tilde{w}_i = 0$ with probability at least $1 - \delta/3$ by Proposition 6.4.11. If $i \in I_{\mathcal{S}}$ then it proceeds to step 3 with probability at least $1 - \delta/3$. At step 3.a, the estimate \tilde{s}_i satisfies $|\tilde{s}_i - \mathbb{E}[S_i]| \leq \mathbb{E}[S_i]/2$ with probability at least $1 - \delta/3$ since $f(\mathbb{E}[S_i]) \geq \sqrt{\text{Var}[S_i]/\mathbb{E}[S_i]} \geq 1$. Finally, at step 3.b, if $|\tilde{s}_i - \mathbb{E}[S_i]| \leq \mathbb{E}[S_i]/2$ and $i \in I_{\mathcal{S}}$ then by Proposition 6.4.15 we have that $\mathcal{V} \geq \sqrt{\frac{10n}{3\mathbb{E}[S_i]}} \geq \sqrt{\frac{10n\nu_i}{9\mathbb{E}[W_i]}} \geq \sqrt{\frac{n\nu_{i+1}}{\mathbb{E}[W_i]}} \geq \frac{\sqrt{\text{Var}[W_i]}}{\mathbb{E}[W_i]} \geq 1$, $\mathbb{E}[W_i] \geq \alpha$ and T_2 is larger than the ℓ_2 -average stopping time of $\text{BucketWeight}(i)$. In this case, the output \tilde{w}_i satisfies $|\tilde{w}_i - \mathbb{E}[W_i]| \leq \frac{\epsilon}{25} \mathbb{E}[W_i]$ with probability at least $1 - \delta/3$. Since $\mathbb{E}[W_i] \in \left[(1 - \frac{\epsilon}{10}) \sum_{v \in B_i} \tau_v, (1 + \frac{\epsilon}{10}) \sum_{v \in \hat{B}_i} \hat{\tau}_v\right]$ by Proposition 6.4.15, we obtain that $\tilde{w}_i \in \left[(1 - \frac{\epsilon}{25})(1 - \frac{\epsilon}{10}) \sum_{v \in B_i} \tau_v, (1 + \frac{\epsilon}{25})(1 + \frac{\epsilon}{10}) \sum_{v \in \hat{B}_i} \hat{\tau}_v\right] \subset \left[(1 - \frac{3\epsilon}{20}) \sum_{v \in B_i} \tau_v, (1 + \frac{3\epsilon}{20}) \sum_{v \in \hat{B}_i} \hat{\tau}_v\right]$. This concludes the proof of parts (1)–(3).

The complexity of step 1 is $O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$ by Proposition 6.4.11. The complexity of step 3.a is $O^*\left(\sqrt{\frac{n}{\mathbb{E}[S_i]}} \left(1 + \frac{m^{3/2}}{n\nu_i}\right)\right)$ with probability at least $1 - \delta/3$ by Proposition 5.4.6. The complexity of step 3.b is $O^*\left(\sqrt{\frac{n}{\mathbb{E}[S_i]}} \left(1 + \frac{m^{3/2}}{n\nu_i} + \frac{\mathbb{E}[S_i]}{n} \frac{m^{3/2}}{t}\right)\right)$ when $|\tilde{s}_i - \mathbb{E}[S_i]| \leq \mathbb{E}[S_i]/2$ by Proposition 6.4.15. Thus, if $i \in I_{\mathcal{S}}$ then the complexity of step 3.b is $O^*\left(\sqrt{\frac{n}{|B_i|}} + \frac{m^{3/2}}{|B_i|\nu_i} + \frac{m^{3/2}}{t}\right) \leq O^*\left(\sqrt{\frac{n\nu_{i+1}}{t}} + \frac{m^{3/2}\nu_{i+1}}{t\nu_i} + \frac{m^{3/2}}{t}\right) \leq O^*\left(\sqrt{\frac{n}{t^{1/3}}} + \frac{m^{3/2}}{t}\right)$ where we used that $\mathbb{E}[S_i] \geq \frac{3}{4}|B_i|$ by Proposition 6.4.10, and $|B_i| \geq \frac{\epsilon t}{30k\nu_{i+1}}$ and $\nu_i \leq \tau$ by definition of $I_{\mathcal{S}}$. This proves part (4) of the proposition.

Finally, for part (5), according to Proposition 6.4.15 and Proposition 6.4.10 the expectation value of the random variable W_i computed by $\text{BucketWeight}(i)$ is at most $\mathbb{E}[W_i] \leq \nu_{i+1}\mathbb{E}[S_i] \leq \nu_{i+1}|\hat{B}_i| + \frac{\epsilon}{180k}$. Moreover, the estimate \tilde{w}_i computed at step 3.b satisfies $\tilde{w}_i \leq 2\mathbb{E}[W_i]$ with probability at least $1 - \delta/3$ by part (2) of Theorem 5.4.4. Thus, $\tilde{w}_i \leq 2\nu_{i+1}|\hat{B}_i| + \frac{\epsilon}{90k}$. \square

6.4.5 Final algorithm

We use the results from the previous sections to describe the final triangle counting algorithm. We first show how to estimate the triangle count t with relative error ϵ when Assumptions 6.A–6.D hold. We later remove these assumptions.

Proposition 6.4.17. *There is a quantum algorithm $\text{ApproxTriangleCount}(\bar{m}, \bar{t}, \tau, (\nu_i^-)_i, (\nu_i^+)_i, (\nu_i)_i, \epsilon, \delta)$ that takes as input an edge estimate \bar{m} , a triangle estimate \bar{t} , a threshold value $\tau > 0$, three sequences $(\nu_i^-)_i, (\nu_i^+)_i, (\nu_i)_i$ and two reals $\epsilon, \delta \in (0, 1/2)$, and that outputs a triangle estimate \tilde{t} such that,*

- (1) *If Assumptions 6.A–6.D hold then $|\tilde{t} - t| \leq \epsilon t$ with probability at least $1 - \delta$.*
- (2) *If Assumptions 6.A and 6.D hold then $\tilde{t} \leq 3t$ with probability at least $1 - \delta$.*
- (3) *The query complexity of the algorithm is $O^*\left(\frac{\sqrt{n}}{\tilde{t}^{1/6}} + \frac{\bar{m}^{3/4}}{\sqrt{\tilde{t}}}\right)$.*

Proof. The algorithm consists of computing an estimate \tilde{w}_i of the weight of the i -th bucket by using the algorithm $\text{ApproxBucketWeight}(i, \delta/(2k))$ (Proposition 6.4.16) for each $i \in [k]$, and to output the triangle estimate $\tilde{t} = \sum_{i \in [k]} \tilde{w}_i$.

According to Proposition 6.4.16, if Assumptions 6.A–6.D hold then $(1 - \frac{3\epsilon}{20}) \sum_{v \in \mathcal{S}} \tau_v \leq \tilde{t} \leq (1 + \frac{3\epsilon}{20}) \sum_{v \in \hat{\mathcal{S}}} \hat{\tau}_v$ with probability at least $(1 - \delta/(2k))^k \geq 1 - \delta/2$. In this case, by Proposition 6.4.6 and Proposition 6.4.7, we have $\tilde{t} \in [(1 - \frac{4\epsilon}{5})(1 - \frac{3\epsilon}{20})t, (1 + \frac{3\epsilon}{5})(1 + \frac{3\epsilon}{20})t] \subset [(1 - \epsilon)t, (1 + \epsilon)t]$. If only Assumptions 6.A and 6.D hold, we have $\tilde{t} = \sum_{i \in [k]} \tilde{w}_i \leq \sum_{i \in [k]} (2\nu_i^+ |\hat{B}_i| + \frac{\epsilon}{90k}) \leq 3t$ with probability at least $1 - \delta/2$, where the first inequality is by part (5) of Proposition 6.4.16, and the second inequality is by Assumption 6.D. This proves parts (1) and (2) of the proposition.

According to Proposition 6.4.16, if Assumptions 6.A–6.D hold and $i \in I_{\mathcal{S}}$ then the query complexity of $\text{ApproxBucketWeight}(i, \delta/(2k))$ is $O^*\left(\frac{\sqrt{n}}{\tilde{t}^{1/6}} + \frac{\bar{m}^{3/4}}{\sqrt{\tilde{t}}}\right)$ with probability at least $1 - \delta/(2k)$. If the computation takes more time than that, then we stop it and we set $\tilde{w}_i = 0$. The new estimate $\tilde{t} = \sum_{i \in [k]} \tilde{w}_i$ is still large enough by part (1) of Proposition 6.4.16. \square

We finally explain how to get rid of the assumptions made in the previous algorithm. Assumptions 6.A, 6.C and 6.D are satisfied with high probability by using the algorithms described in Section 6.3, Lemma 6.4.1 and Lemma 6.4.2 respectively. Assumption 6.B is handled by running the algorithm from Proposition 6.4.17 over a decreasing sequence $\bar{t} = n^3, n^3/2, n^3/4, \dots$. We use part (2) of Proposition 6.4.17 to stop this sequence when \bar{t} satisfies Assumption 6.B with high probability. The final result is given in Algorithm 6.9.

Theorem 6.4.18 (TRIANGLE COUNTING). *Let G be a graph with n vertices, m edges and t triangles in the quantum general graph model, and fix an error parameter $\epsilon \in (0, 1/4)$. Then, the triangle counting algorithm (Algorithm 6.9) outputs a triangle estimate \tilde{t} such that $|\tilde{t} - t| \leq \epsilon t$ and it uses $O^*\left(\frac{\sqrt{n}}{\tilde{t}^{1/6}} + \frac{m^{3/4}}{\sqrt{\tilde{t}}}\right)$ quantum queries with probability at least $1 - \frac{1}{\log n}$.*

Proof. By a union bound, Assumptions 6.A, 6.C and 6.D hold at each step of the algorithm with probability at least $1 - \frac{1}{5 \log n} - \frac{3 \log n}{30 \log^2(n)} - \frac{1}{2 \log n} = 1 - \frac{4}{5 \log n}$. Let us assume that they indeed hold. Consider the estimate \tilde{t} computed at step 3.b. If $\bar{t} > t$ then $\tilde{t} \leq 3t < 3\bar{t}$ with probability at least $1 - \frac{1}{15 \log^2 n}$ by part (2) of Proposition 6.4.17. If $\bar{t} \in [t/8, t]$ (i.e. Assumption 6.B holds) then $|\tilde{t} - t| \leq \epsilon t$ by part (1) of Proposition 6.4.17. Furthermore, if $\bar{t} \in [t/8, t/4]$ and $|\tilde{t} - t| \leq \epsilon t$ then $\tilde{t} \geq (1 - \epsilon)t \geq 3t/4 \geq 3\bar{t}$. Consequently, by a union

1. Compute an edge estimate \tilde{m} by using the algorithm of Theorem 6.3.2 with error $1/2$ and failure probability $\frac{1}{5 \log n}$. Set $\bar{m} = \tilde{m}/2$.
2. Compute the buckets' boundaries $(\nu_i^-)_i, (\nu_i^+)_i, (\nu_i)_i$ by using Lemma 6.4.2.
3. For $\bar{t} = n^3, n^3/2, n^3/4, \dots, 1$:
 - a) Compute a threshold value τ by using Lemma 6.4.1.
 - b) Compute a triangle count estimate \tilde{t} by using the algorithm `Approx-TriangleCount` $\left(\bar{m}, \bar{t}, \tau, (\nu_i^-)_i, (\nu_i^+)_i, (\nu_i)_i, \epsilon, \frac{1}{15 \log^2 n}\right)$ from Proposition 6.4.17.
 - c) If $\tilde{t} \geq 3\bar{t}$ then stop and output \tilde{t} .
4. Output $\tilde{t} = 1$.

Algorithm 6.9: Triangle counting algorithm.

bound over the (at most) $3 \log(n)$ iterations of step 4, the algorithm stops when $\bar{t} \in [t/8, t]$ and outputs a value \tilde{t} such that $|\tilde{t} - t| \leq \epsilon t$ with probability at least $1 - \frac{1}{5 \log n}$. In this case, the total query complexity is $O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$ by Proposition 6.4.17. \square

6.5 Lower bounds

We prove two lower bounds for the Edge and Triangle Counting problems by using a standard reduction technique from the two-player communication problem Disjointness. The reductions are based on two families of graphs constructed in [ER18], whose main properties are given below. We first describe the general reduction framework. We define an *efficient graph embedding* as a mapping from an input pair (x, y) in the communication model to a graph $\mathcal{E}(x, y)$ such that any query in the general graph model can be answered by an efficient communication protocol.

Definition 6.5.1. Let $N, n > 0$ be two integers. Consider a set $P \subseteq \{0, 1\}^N \times \{0, 1\}^N$ and a function $\mathcal{E} : P \rightarrow \mathcal{G}_n$. We say that \mathcal{E} is an *efficient graph embedding* if it satisfies the following condition. For any degree, neighbor or edge query, there exists a zero-error two-party classical communication protocol with communication cost $O(1)$ such that for any $(x, y) \in P$ the protocol computes the result of the given query on the graph $\mathcal{E}(x, y)$.

Eden and Rosenbaum [ER18] described an efficient graph embedding that relates the Disjointness problem to the number of r -cliques in a graph. The Disjointness function maps an input $(x, y) \in \{0, 1\}^N \times \{0, 1\}^N$ to 0 if there exists an index $i \in [N]$ such that $x_i = y_i = 1$, and to 1 otherwise.

Proposition 6.5.2 (Theorem 4.1 and Corollary 4.3 in [ER18]). *Let r be a fixed constant. Let $n, C_r > 0$ be two integers such that $C_r \leq \binom{n/2}{r}$ and let $\epsilon \in (0, 1/3)$. Set $N = \left\lfloor \frac{cn}{(\epsilon C_r)^{1/r}} \right\rfloor$ where c is a universal constant. Then, there exists an efficient graph embedding $\mathcal{E} : \{0, 1\}^N \times \{0, 1\}^N \rightarrow \mathcal{G}_n$ such that,*

- (1) *If $\text{Disjointness}(x, y) = 1$ then the number of r -cliques in $\mathcal{E}(x, y)$ is equal to C_r .*
- (2) *If $\text{Disjointness}(x, y) = 0$ then the number of r -cliques in $\mathcal{E}(x, y)$ is equal to $(1 + 3\epsilon)C_r$.*

We use the above result to first show that the edge counting algorithm (Theorem 6.3.2) is nearly optimal. We use a standard technique developed in [BCW98] to simulate a query algorithm with a communication protocol.

Proposition 6.5.3. *Any algorithm that estimates the number m of edges in an n -vertex graph with relative error $\epsilon \in (0, 1)$ and success probability $2/3$ must perform at least $\Omega\left(\frac{\sqrt{n}}{(\epsilon m)^{1/4} \log(n)}\right)$ quantum queries in the general graph model.*

Proof. Fix n, m, ϵ and consider any quantum query algorithm that can estimate the number of edges with relative error ϵ and success probability $2/3$. Let $q(n, m, \epsilon)$ denote the maximum number of quantum queries performed by this algorithm over all input graphs with n vertices and m or $(1 + 3\epsilon)m$ edges. We transform this algorithm into a communication protocol with cost $O(q(n, m, \epsilon) \log(n))$ for solving Disjointness on input size $N = \left\lfloor \frac{cn}{\sqrt{\epsilon m}} \right\rfloor$ as follows. Consider an input $(x, y) \in \{0, 1\}^N \times \{0, 1\}^N$ distributed over two players Alice and Bob. Alice simulates the quantum edge counting algorithm on the graph $\mathcal{E}(x, y)$ defined by the efficient graph embedding of Proposition 6.5.2. Each time a quantum query must be performed, Alice appends an ancilla register of $O(1)$ qubits to the query register (made of $O(\log n)$ qubits), and she exchanges the two registers with Bob over $O(1)$ rounds of communication to simulate in superposition the protocol provided by Definition 6.5.1. The simulation uses the same technique as in [BCW98, Theorem 2.1]. Once Alice has computed an edge estimate with relative error ϵ , she can distinguish between the graph having m or $(1 + 3\epsilon)m$ edges. According to Proposition 6.5.2, it allows her to obtain the value of $\text{Disjointness}(x, y)$ with success probability $2/3$. Since the quantum communication complexity of Disjointness is $\Omega(\sqrt{N})$ [Raz03], we conclude that $O(q(n, m, \epsilon) \log(n)) \geq \Omega(\sqrt{N})$. \square

Next, we present an efficient graph embedding that relates the k -Intersection problem to the number of triangles in a graph. The Intersection_k function maps an input $(x, y) \in \{0, 1\}^N \times \{0, 1\}^N$ to 1 if there exists at least k indices $i \in [N]$ such that $x_i = y_i = 1$, and to 0 if there is no index $i \in [N]$ such that $x_i = y_i = 1$. The function is undefined otherwise.

Proposition 6.5.4 (Theorem 4.7 in [ER18]). *Let $n, m, t > 0$ be three integers such that $m \leq n^2/9$ and $t \leq m^{3/2}/8$. Set $N = m/4$ and $k = \max(1, 2t/\sqrt{m})$. If N and k are integers, then there is an efficient graph embedding $\mathcal{E} : \{0, 1\}^N \times \{0, 1\}^N \rightarrow \mathcal{G}_n$ such that,*

- (1) *If $\text{Intersection}_k(x, y) = 1$ then $\mathcal{E}(x, y)$ contains m edges and no triangle.*
- (2) *If $\text{Intersection}_k(x, y) = 0$ then $\mathcal{E}(x, y)$ contains m edges and at least t triangles.*

We finally use the two previous graph embeddings to obtain the next lower for the Triangle Counting problem.

Proposition 6.5.5. *Any algorithm that estimates the number t of triangles in an n -vertex m -edge graph with relative error $\epsilon = 1/2$ and success probability $2/3$ must perform at least $\Omega\left(\frac{1}{\log n} \left(\frac{\sqrt{n}}{t^{1/6}} + \min\left(\frac{m^{3/4}}{\sqrt{t}}, \sqrt{m}\right)\right)\right)$ quantum queries in the general graph model.*

Proof. The proof is similar to that of Proposition 6.5.3. The two terms in the lower bound are proved separately by using the efficient graph embeddings of Proposition 6.5.2 (with $r = 3$) and Proposition 6.5.4 respectively. The second term in the lower bound uses that the quantum communication complexity of the k -Intersection problem is $\Omega(\sqrt{N/k})$, which can be proved by a simple reduction from Disjointness (see [ER18, Corollary 2.7] for instance). \square

6.6 Discussion

There is a gap in our understanding of the triangle counting problem in the small triangle count regime where $t \leq O(m)$. The running time of our algorithm is $O^*\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right)$ (Theorem 6.4.18), whereas the best known query lower bound is $\tilde{\Omega}\left(\frac{\sqrt{n}}{t^{1/6}} + \sqrt{m}\right)$ (Proposition 6.5.5) in this case. Classically [ELR15; Ses15; ELRS17; ERS20b; ERS20a], the optimal *query* complexity is $O^*\left(\frac{n}{t^{1/3}} + m\right)$ when $t \leq O(m)$, although the best known *gate* complexity is $O^*\left(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}\right)$. The improvement in terms of query complexity comes from keeping the results of all queries in memory so that the same edge is never queried twice. We do not know if a similar improvement is possible in the quantum query model. We suggest to first study the problem of *finding* one triangle in the quantum general graph model. A simple adaptation of our triangle counting algorithm leads to a query complexity of $\tilde{O}(\sqrt{n} + m^{3/4})$, whereas one can show a lower bound of $\tilde{\Omega}(\sqrt{n} + \sqrt{m})$.

We suspect that the techniques used in this chapter can be extended to counting the number of k -cliques in a graph [ERS20b; ERS20a], and to approximate other graph parameters such as the size of a minimum spanning tree [CRT05]. Another improvement could be to express the quantum query complexity of these problems in terms of *graph arboricity* as in [ERS20b; ERS20a].

Part III

Quantum Algorithms for Optimization with Importance Sampling

7

Quantum State Preparation and Importance Sampling

Section 7.5 of this chapter is based on the following paper:

[HRRS19] Y. Hamoudi, P. Reberntrost, A. Rosmanis, and M. Santha. “Quantum and Classical Algorithms for Approximate Submodular Function Minimization”. In: *Quantum Information & Computation* 19.15&16 (2019), pp. 1325–1349.

7.1 Introduction

The preparation of a specific quantum state is an important building block and a critical bottleneck in many quantum algorithms [MNRS11; HHL09; Ber14; Aar15]. The objective of the *State Preparation* problem is to find the minimum amount of resources needed to generate a quantum state given some description of it. In general, the complexity of this problem scales linearly with the dimension of the state to be prepared [MVBS05; PB11]. Yet, it is possible to achieve sublinear bounds for particular states or input models. One such example is the *black-box model* where, given oracle access to a non-zero real vector $w = (w_1, \dots, w_N)$, the objective is to load the associated normalized probability vector into the amplitudes of the $\log(N)$ -qubit state $|w\rangle$ defined as

$$|w\rangle = \frac{1}{\sqrt{\|w\|_1}} \sum_{i \in [N]} \sqrt{w_i} |i\rangle.$$

Grover adapted his celebrated quantum search algorithm to this problem in [Gro00b], where he showed that $O(\sqrt{N})$ queries to w are sufficient to prepare $|w\rangle$. In practice, one can expect that *several* copies of the same quantum state are needed for further use. For instance, $|w\rangle$ may be fed in an algorithm that fails with some probability and that must be repeated several times. The no-cloning theorem prevents the state $|w\rangle$ from being easily duplicated. In fact, it is easy to show that additional queries to the input are required to prepare several copies of $|w\rangle$. The problem of adapting the state preparation procedure to the desired number K of copies has received little attention. Usually, it is possible to simply repeat K times the procedure used to prepare one copy, but the complexity grows linearly with K . For instance, the algorithm of Grover leads to a query complexity of $O(K\sqrt{N})$ for preparing the K -fold state $|w\rangle^{\otimes K}$. In this chapter, we investigate the question of whether a more efficient algorithm exists. We describe a preprocessing phase that uses $O(\sqrt{KN})$ queries, after which each copy of $|w\rangle$ requires $O(\sqrt{N/K})$ queries to be prepared. Our result improves upon the previous approach by a factor of \sqrt{K} , and it is shown to be optimal.

The State Preparation problem is intimately related to the task of preparing samples from a discrete distribution. In the *Importance Sampling* problem, the objective is to generate K independent samples from the probability vector $\left(\frac{|w_1|}{\|w\|_1}, \dots, \frac{|w_N|}{\|w\|_1}\right)$ associated with a weight vector (w_1, \dots, w_N) . There is a rich literature on importance sampling, which dates back to the early days of the Monte Carlo method and to the work of Von Neuman on random number generation [Neu51]. It has found numerous applications in computer simulations, statistics, numerical analysis, etc. The *Born rule* paves the way for a quantum mechanical approach to random number generation. Grover [Gro00a; Gro00b] suggested addressing the Importance Sampling problem by preparing the state $|w\rangle$ and measuring it in the computational basis. We use our state preparation algorithm for generating K independent samples faster than it is possible classically. We also propose an alternative and more direct approach that does not require preparing the full quantum state $|w\rangle^{\otimes K}$. These results are applied in the next chapter to obtain a quantum speedup for two stochastic optimization methods (multiplicative weight update and stochastic gradient descent).

7.1.1 Related work

Our work is based on the *Quantum Rejection Sampling* method, where a state that is easy to prepare (e.g. a uniform superposition) is mapped to a target state by amplitude amplification. This method was pioneered by Grover in [Gro00b] (Proposition 7.3.2) and subsequently studied in [ORR13; SLSB19; LYC14; WG17]. All of these works (except for [LYC14]) take place in the quantum oracle model and they often require a number of queries that is polynomial in the dimension N of the state. In the non-oracular setting, the problem of loading an arbitrary vector (w_1, \dots, w_N) into the amplitudes of a quantum state requires a circuit of depth $O(N)$ and width $O(\log N)$ [MVBS05; PB11]. It is possible to use only $\text{polylog}(N)$ resources for specific cases such as efficiently integrable probability distributions [Zal98; GR02; KM01] (Proposition 7.3.3), uniformly bounded amplitudes [SS06], Gaussian states [KW09] or probability distributions resulting from a Bayesian network [LYC14]. A different line of work [AT07; SBBK08; WA08; OBD18; Ape19; HW20] studied the preparation of a quantum state that corresponds to the stationary distribution of a Markov chain. These algorithms use Markov chain Monte Carlo methods and quantum walk techniques to obtain a preparation time scaling with the spectral gap. Aharonov and Ta-Shma [AT07] also showed that the existence of an efficient procedure to convert any circuit into a coherent state encoding the output distribution of that circuit would imply that $\text{SZK} \subseteq \text{BQP}$.

We refer the reader to [Dev86; BFS87; Knu98] for a general overview of classical techniques for sampling from a discrete distribution. In the Importance Sampling problem that we consider here (also called Weighted Sampling or L_1 Sampling), given a non-zero vector $w = (w_1, \dots, w_N)$, the objective is to construct a data structure that supports fast sampling of an element $i \in [N]$ with probability proportional to $|w_i|$. The *alias method* [Wal74; Wal77; KP79; Vos91] solves this problem with a preprocessing cost of $O(N)$ operations, and a generating cost of $O(1)$ operations per sample (Proposition 7.5.2). This method was subsequently adapted to other settings such as dynamic weight updates [HMM93; MVN03], subset sampling [BP17] and space-efficient data structures [BL13; BIJK18]. Knuth and Yao [KY76] initiated the study of entropy-efficient samplers that ought to minimize the number of random bits used for generating one sample. The related question of minimizing the number of non-basis-preserving gates for quantum state preparation was studied by Shi [Shi05].

7.1.2 Contributions and organization

We first present in Section 7.2 the input model used in this chapter. Next, we recall in Section 7.3 some algorithms for maximum finding and quantum state preparation that are needed for our work. We study the problem of preparing K copies of a quantum state in Section 7.4. Our first main result is summarized in the next theorem.

Theorem 7.4.3 (Restated). *There is a quantum algorithm with the following properties. Consider two integers $1 \leq K \leq N$, a real $\delta \in (0, 1)$ and a non-zero vector $w \in \mathbb{R}^N$. Then, with probability at least $1 - \delta$, the algorithm outputs K copies of the state $|w\rangle = \frac{1}{\sqrt{\|w\|_1}} \sum_{i \in [N]} \sqrt{|w_i|} |i\rangle$ and it performs $O(\sqrt{KN} \log(1/\delta))$ queries to w in expectation.*

We study the problem of sampling K elements from a discrete probability distribution in Section 7.5. We obtain the next result as a simple corollary of Theorem 7.4.3.

Theorem 7.5.1 (Restated). *There is a quantum algorithm with the following properties. Consider two integers $1 \leq K \leq N$, a real $\delta \in (0, 1)$ and a non-zero vector $w \in \mathbb{R}^N$. Then, with probability at least $1 - \delta$, the algorithm outputs K independent samples $i_1, \dots, i_K \in [N]$ from the probability distribution $\mathcal{D}_w = (\frac{|w_1|}{\|w\|_1}, \dots, \frac{|w_N|}{\|w\|_1})$ on $[N]$ and it performs $O(\sqrt{KN} \log(1/\delta))$ queries to w in expectation.*

We propose a simpler algorithm for the above problem when w is of unit norm (Theorem 7.5.3). We finally show that our results are nearly optimal in Proposition 7.5.4.

7.1.3 Proof overview

K -fold state preparation. Our starting point is the result from Grover [Gro00b] (presented in Proposition 7.3.2) for preparing one copy of the state $|w\rangle$ in the black-box model. This algorithm uses an upper bound m on the largest value $\max_i |w_i|$ to construct a unitary algorithm U such that,

$$U|0\rangle = \frac{1}{\sqrt{N}} \sum_{i \in [N]} |i\rangle \left(\sqrt{1 - \frac{|w_i|}{m}} |0\rangle + \sqrt{\frac{|w_i|}{m}} |1\rangle \right).$$

The state $|w\rangle|1\rangle$ has amplitude $\sqrt{\frac{\|w\|_1}{Nm}}$ in $U|0\rangle$, thus it can be extracted by using the amplitude amplification algorithm with $O\left(\sqrt{\frac{Nm}{\|w\|_1}}\right)$ applications of U and U^{-1} . In particular, if the largest coordinate of w (in absolute value) is smaller than $m = \|w\|_1/K$ then we can prepare one copy of $|w\rangle$ in time $O(\sqrt{N/K})$, and K copies in time $O(\sqrt{KN})$. We use this observation to construct a new circuit \mathcal{C} such that $|w\rangle|1\rangle$ has amplitude at least $K/(2N)$ in $\mathcal{C}|0\rangle$, even if w contains large coordinates. The circuit \mathcal{C} uses only two queries to w , but it requires $O(\sqrt{KN})$ queries to be constructed during a preprocessing phase that is executed only once (Proposition 7.4.1). The preprocessing phase first computes the set H that contains the positions of the K largest coordinates in w , by using a variant of the quantum maximum finding algorithm (Proposition 7.3.1). The circuit \mathcal{C} proceeds in two stages. First, it prepares a state whose amplitudes depend only on the values in $\{w_i : i \in H\}$ (step 4 of Algorithm 7.2). Next, it modifies this state by querying the set $\{w_i : i \notin H\}$ in a way that is similar to that of U . The crucial observation is that the values in $\{|w_i| : i \notin H\}$ must be smaller than $\|w\|_1/K$ by definition of H , thus they can be amplified at a smaller cost. Finally, each copy of $|w\rangle$ can be obtained by one application of the amplitude amplification algorithm on \mathcal{C} (Proposition 7.4.2).

K -fold importance sampling. We propose two different algorithms for sampling K independent elements from the distribution $\mathcal{D}_w = (\frac{|w_1|}{\|w\|_1}, \dots, \frac{|w_N|}{\|w\|_1})$ given query access to w . The first algorithm (Theorem 7.5.1) prepares K copies of $|w\rangle$ and it measures them in the computational basis. The second algorithm (Theorem 7.5.3) does not need to prepare the state $|w\rangle$, but it requires that $\|w\|_1 = 1$. We sketch this second algorithm. The first step is to compute the same set H as before. Next, we load in time $O(K)$ the conditional distribution $(\frac{|w_i|}{\|w_H\|_1})_{i \in H}$, where $\|w_H\|_1 = \sum_{i \in H} |w_i|$, into a classical data structure [Wal77; KP79; Vos91] (Proposition 7.5.2) that supports fast sampling in time $O(1)$. The complement distribution $(\frac{|w_i|}{1 - \|w_H\|_1})_{i \notin H}$ will be sampled from differently, by preparing the corresponding quantum state $\frac{1}{\sqrt{1 - \|w_H\|_1}} \sum_{i \notin H} \sqrt{|w_i|} |i\rangle$ with the algorithm from Grover [Gro00b] described before in time $O\left(\sqrt{\frac{N \cdot \max_{i \notin H} |w_i|}{1 - \|w_H\|_1}}\right) = O\left(\sqrt{\frac{N}{K(1 - \|w_H\|_1)}}\right)$. Each of the K samples is obtained by first flipping a coin that lands head with probability $\|w_H\|_1$, and then sampling $i \in H$ from the classical data structure (head case) or $i \notin H$ by quantum state preparation (tail case). The head case occurs $K\|w_H\|_1$ times in expectation. Thus, the total expected time is $O\left(K\|w_H\|_1 \cdot 1 + K(1 - \|w_H\|_1) \cdot \sqrt{\frac{N}{K(1 - \|w_H\|_1)}}\right) = O(\sqrt{KN})$.

7.2 Model of input

We assume for simplicity that N is a power of two in this chapter. The input to the problem is a non-zero vector $w \in \mathbb{R}^N$ with oracle access to its coordinates. We suppose that all the real numbers manipulated by our algorithms can be encoded over s bits, for a fixed value of s . The oracle is represented as a unitary operator \mathcal{O}_w such that $\mathcal{O}_w(|i\rangle|v\rangle) = |i\rangle|v \oplus w_i\rangle$ for all $i \in [N]$ and $v \in \{0, 1\}^s$. We associate with each vector $w \in \mathbb{R}^N$ a quantum state $|w\rangle$ and a distribution \mathcal{D}_w defined as follows.

Definition 7.2.1. Given a non-zero vector $w \in \mathbb{R}^N$, define the state $|w\rangle = \sum_{i=1}^N \sqrt{\frac{|w_i|}{\|w\|_1}} |i\rangle$ and the distribution $\mathcal{D}_w = (\frac{|w_1|}{\|w\|_1}, \dots, \frac{|w_N|}{\|w\|_1})$ that gives $i \in [N]$ with probability $\frac{|w_i|}{\|w\|_1}$.

We use the quantum circuit model over a universal gate set made of the CNOT gate and of all one-qubit gates. We augment it with a set of arithmetic gates that perform standard operations over basis states, such as the addition $|w_1\rangle|w_2\rangle|0\rangle \mapsto |w_1\rangle|w_2\rangle|w_1 + w_2\rangle$. We also add the three gates described in Figure 7.1. The indicator gate $\mathbb{1}_H$ is specified by a subset $H \subset [N]$. It operates on a Boolean value b and an index $i \in [N]$. The query gate \mathcal{O}_w is specified by the input vector w to the problem. It operates on an index $i \in [N]$ and a real v (encoded over s bits). Finally, the controlled rotation gate R_m is specified by a real $m > 0$ and it operates on a Boolean value b and a real v . We refer the reader to [HRS18; SLSB19] and references therein for efficient implementations of the arithmetic gates and controlled rotation gates.

7.3 Preliminaries

The next algorithm generalizes the quantum minimum finding [DH96] to finding K largest entries in a vector w . The algorithm succeeds if it outputs a set $H \subset [N]$ of size K such that for all $i \notin H$ and $j \in H$ we have $|w_i| \leq |w_j|$. There is not necessarily a unique choice for H since different coordinates of w may be equal.

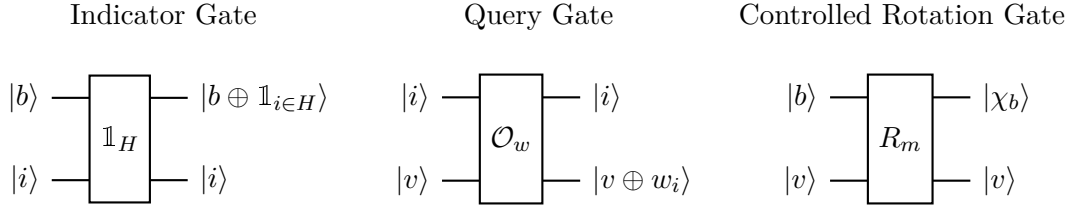


Figure 7.1: Three allowed gates. The state $|\chi_b\rangle$ is defined as $\sqrt{1 - \frac{|v|}{m}}|b\rangle + (-1)^b \sqrt{\frac{|v|}{m}}|1-b\rangle$ if $|v| \leq m$, and $|b\rangle$ otherwise.

Proposition 7.3.1 (TOP- K MAXIMUM FINDING – Theorem 4.2 in [DHHM06]). *There exists a quantum algorithm with the following properties. Consider two integers $1 \leq K \leq N$, a real $\delta \in (0, 1)$ and a vector $w \in \mathbb{R}^N$. Then, the algorithm outputs the positions of K largest entries in w (in absolute value) with success probability at least $1 - \delta$, and it performs $O(\sqrt{KN} \log(1/\delta))$ queries to w .*

The next result is a quantum state preparation algorithm from Grover [Gro00b] that takes as input an upper bound m on the largest coordinate in w . The optimal value $m = \max_{i \in [N]} |w_i|$ can be computed in time $O(\sqrt{N})$ by using the maximum finding algorithm, in which case the next algorithm uses $O(\sqrt{N})$ queries too. We can measure the state $|w\rangle$ in the computational basis to obtain one sample from the distribution \mathcal{D}_w . The main contribution of this chapter is to extend this result to the problem of preparing K copies of $|w\rangle$ and sampling K independent samples from \mathcal{D}_w .

Proposition 7.3.2 (STATE PREPARATION & IMPORTANCE SAMPLING – [Gro00b]). *There is a quantum algorithm with the following properties. Consider an integer N , a non-zero vector $w \in \mathbb{R}^N$ and a real m such that $m \geq \max_i |w_i|$. Then, the algorithm outputs the state $|w\rangle$ with probability at least $1 - \delta$. Alternatively, the algorithm can output one sample from the distribution \mathcal{D}_w . The algorithm performs $O\left(\sqrt{\frac{Nm}{\|w\|_1}}\right)$ queries to w in expectation.*

We present a second state preparation algorithm that requires having an efficient procedure to compute the value of $\sum_{\ell=i}^j |w_\ell|$ for any $1 \leq i \leq j \leq N$.

Proposition 7.3.3 (STATE PREPARATION BY INTEGRATION – [Zal98; KM01; GR02]). *There is a quantum algorithm with the following properties. Consider an integer N and a non-zero vector $w \in \mathbb{R}^N$ such that there is a reversible circuit with T gates that computes $\sum_{\ell=i}^j |w_\ell|^2$ given $i \leq j$. Then, the algorithm outputs $|w\rangle$ and it uses $O(T \log N)$ gates.*

7.4 Preparing K copies of a quantum state

We describe our state preparation algorithm for preparing K copies of $|w\rangle$ given oracle access to $w = (w_1, \dots, w_N)$. The first step of the algorithm is a preprocessing phase that constructs a particular circuit \mathcal{C} described in Algorithm 7.2.

Proposition 7.4.1 (PREPROCESSING PHASE). *Consider two integers $1 \leq K \leq N$, a real $\delta \in (0, 1)$ and a non-zero vector $w \in \mathbb{R}^N$. Then, Algorithm 7.2 outputs with probability at least $1 - \delta$ the description of a unitary circuit \mathcal{C} that prepares the state*

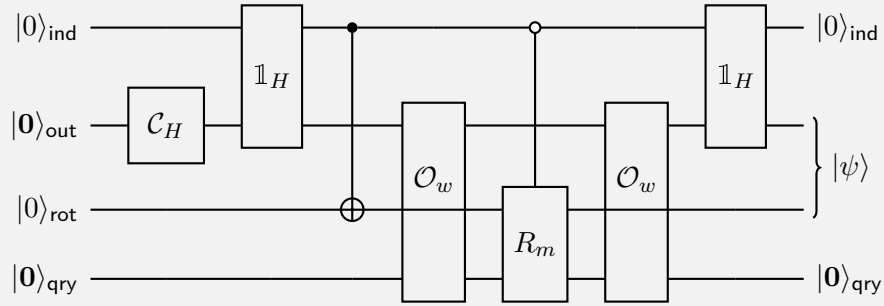
$$|\psi\rangle = \sqrt{1 - \alpha_w} |w^\perp\rangle |0\rangle + \sqrt{\alpha_w} |w\rangle |1\rangle$$

where $\alpha_w \geq K/(2N)$ and $|w^\perp\rangle$ is a unit state. The algorithm performs $O(\sqrt{KN} \log(1/\delta))$ queries to w . The circuit \mathcal{C} performs 2 queries to w and it uses $O(K \log N)$ gates.

1. Compute a set $H \subset [N]$ of the positions of K largest entries in w (in absolute value) by using the **top- K maximum finding** algorithm with failure probability δ .
2. Query and store the value of w_i for all $i \in H$.
3. Compute $m = \min_{i \in H} |w_i|$ and $W = (N - K)m + \sum_{i \in H} |w_i|$.
4. Use the state preparation algorithm of Proposition 7.3.3 to construct a circuit \mathcal{C}_H such that, on input $|0\rangle_{\text{out}}$, it prepares the state

$$\mathcal{C}_H |0\rangle_{\text{out}} = \sum_{i \in H} \sqrt{\frac{|w_i|}{W}} |i\rangle_{\text{out}} + \sum_{i \notin H} \sqrt{\frac{m}{W}} |i\rangle_{\text{out}}.$$

5. Construct the following circuit \mathcal{C} (explained in the proof of Proposition 7.4.1),



6. Output \mathcal{C} .

Algorithm 7.2: Preprocessing phase.

Proof. We assume that the top- K maximum finding algorithm returns a correct set H at step 1, which is the case with probability at least $1 - \delta$. The circuit \mathcal{C} defined at step 5 operates on four registers: **ind** and **rot** that contain a Boolean value, **out** that contains an integer $i \in [N]$, and **qry** that contains a real $v \in \mathbb{R}$. The indicator and CNOT gates flip the content of **rot** when the **out** register contains $i \in H$, whereas the rotation gate R_m is activated when $i \notin H$ (the white circle is used to denote the controlled operation $|1\rangle\langle 1| \otimes I + |0\rangle\langle 0| \otimes R_m$). The gates 1_H and \mathcal{O}_w are applied a second time at the end of \mathcal{C} to uncompute the registers **ind** and **qry**. A simple calculation shows that the final state is equal to $|0\rangle_{\text{ind}} |\psi\rangle |0\rangle_{\text{qry}}$ where

$$\begin{aligned} |\psi\rangle &= \sum_{i \in H} \sqrt{\frac{|w_i|}{W}} |i\rangle_{\text{out}} |1\rangle_{\text{rot}} + \sum_{i \notin H} \sqrt{\frac{m}{W}} |i\rangle_{\text{out}} \left(\sqrt{1 - \frac{|w_i|}{m}} |0\rangle_{\text{rot}} + \sqrt{\frac{|w_i|}{m}} |1\rangle_{\text{rot}} \right) \\ &= \sqrt{1 - \frac{\|w\|_1}{W}} |w^\perp\rangle_{\text{out}} |0\rangle_{\text{rot}} + \sqrt{\frac{\|w\|_1}{W}} |w\rangle_{\text{out}} |1\rangle_{\text{rot}} \end{aligned}$$

for some unit state $|w^\perp\rangle_{\text{out}}$ and $|w\rangle_{\text{out}} = \frac{1}{\sqrt{\|w\|_1}} \sum_{i \in [N]} \sqrt{|w_i|} |i\rangle_{\text{out}}$. In order to lower bound the coefficient $\alpha_w = \frac{\|w\|_1}{W}$, we first observe that the smallest value $m = \min_{i \in H} |w_i|$ over H must satisfy $m \leq \frac{\|w\|_1}{K}$ since otherwise $\sum_{i \in H} |w_i|$ would exceed $\|w\|_1$. Thus, $\alpha_w^{-1} = \frac{W}{\|w\|_1} = \frac{(N-K)m + \sum_{i \in H} |w_i|}{\|w\|_1} \leq \frac{N}{K} + 1$. The algorithm performs $O(\sqrt{KN} \log(1/\delta))$

7.5 Preparing K samples from a discrete distribution

queries to w at step 1 according to Proposition 7.3.1, and K queries at step 2. Finally, the circuit \mathcal{C}_H can be constructed with $O(K \log N)$ gates since for any two values $i \leq j$ the sum of the squared coefficients in front of the basis states $|\ell\rangle_{\text{out}}$ for $i \leq \ell \leq j$ at step 4 is easily obtained as $\frac{1}{W} (m \cdot |\{i, \dots, j\} \setminus H| + \sum_{\ell \in H \cap \{i, \dots, j\}} |w_i|)$. \square

We use the circuit \mathcal{C} constructed during the preprocessing phase, together with the amplitude amplification algorithm, to obtain the next state preparation phase that generates one copy of $|w\rangle$ in time $O(\sqrt{N/K})$.

1. Define the projector $\Pi = I \otimes |1\rangle\langle 1|_{\text{rot}}$ acting on the same registers as the circuit \mathcal{C} constructed by Algorithm 7.2 on input K, δ, w .
2. Compute the state $\frac{1}{\|\Pi\mathcal{C}|\mathbf{0}\rangle\|} \Pi\mathcal{C}|\mathbf{0}\rangle$ by using the **sequential amplitude amplification** algorithm $\text{Seq-AAmp}(\mathcal{C}, \Pi)$.
3. Output the state contained in the out register of $\frac{1}{\|\Pi\mathcal{C}|\mathbf{0}\rangle\|} \Pi\mathcal{C}|\mathbf{0}\rangle$.

Algorithm 7.3: State preparation phase.

Proposition 7.4.2 (STATE PREPARATION PHASE). *Consider two integers $1 \leq K \leq N$, a real $\delta \in (0, 1)$ and a non-zero vector $w \in \mathbb{R}^N$. Let \mathcal{C} denote the unitary circuit obtained with Algorithm 7.2 on input K, δ, w . If \mathcal{C} correctly prepares the state $|\psi\rangle$ described in Proposition 7.4.1, then Algorithm 7.3 outputs the state $|w\rangle$ and it performs $O(\sqrt{N/K})$ queries to w in expectation.*

Proof. If the circuit \mathcal{C} prepares the state $|\psi\rangle$ described in Proposition 7.4.1, then $\Pi\mathcal{C}|\mathbf{0}\rangle = \sqrt{\alpha_w} |\mathbf{0}\rangle_{\text{ind}} |w\rangle_{\text{out}} |1\rangle_{\text{rot}} |\mathbf{0}\rangle_{\text{qry}}$. Thus, by Theorem 3.2.3, the sequential amplitude amplification algorithm outputs the state $|\mathbf{0}\rangle_{\text{ind}} |w\rangle_{\text{out}} |1\rangle_{\text{rot}} |\mathbf{0}\rangle_{\text{qry}}$ and it uses $O(\alpha_w^{-1/2}) \leq O(\sqrt{N/K})$ applications of $\mathcal{C}, \mathcal{C}^{-1}$ and $I - 2\Pi$ in expectation. \square

We can finally prepare the state $|w\rangle^{\otimes K}$ by using the two previous algorithms.

Theorem 7.4.3 (K -FOLD STATE PREPARATION). *There exists a quantum algorithm with the following properties. Consider two integers $1 \leq K \leq N$, a real $\delta \in (0, 1)$ and a non-zero vector $w \in \mathbb{R}^N$. Then, with probability at least $1 - \delta$, the algorithm outputs K copies of the state $|w\rangle$ and it performs $O(\sqrt{KN} \log(1/\delta))$ queries to w in expectation.*

Proof. The algorithm consists of applying the preprocessing phase described in Proposition 7.4.1 only once, and then repeating K times the preparation phase of Proposition 7.4.2. The query complexity is $O(\sqrt{KN} \log(1/\delta)) + K \cdot O(\sqrt{N/K}) = O(\sqrt{KN} \log(1/\delta))$. \square

7.5 Preparing K samples from a discrete distribution

We describe two algorithms for generating K independent samples from the distribution \mathcal{D}_w . The first algorithm uses the state preparation of $|w\rangle^{\otimes K}$ described in the previous section.

Theorem 7.5.1 (K -FOLD IMPORTANCE SAMPLING). *There exists a quantum algorithm with the following properties. Consider two integers $1 \leq K \leq N$, a real $\delta \in (0, 1)$ and a non-zero vector $w \in \mathbb{R}^N$. Then, with probability at least $1 - \delta$, the algorithm outputs K independent samples $i_1, \dots, i_K \in [N]$ from the probability distribution \mathcal{D}_w and it performs $O(\sqrt{KN} \log(1/\delta))$ queries to w in expectation.*

Proof. The algorithm consists of preparing K copies of the state $|w\rangle$ with success probability at least $1 - \delta$ by using the algorithm of Theorem 7.4.3, and measuring them in the computational basis. \square

We describe an alternative algorithm that uses a combination of the standard state preparation method of Grover (Proposition 7.3.2) with the classical alias method.

Proposition 7.5.2 (ALIAS METHOD [Wal77; KP79; Vos91]). *There exists a two-phase classical algorithm with the following properties. Given a non-zero vector $w \in \mathbb{R}^N$, it constructs a data structure in time $O(N)$ during the preprocessing phase, and it outputs a sample from the distribution \mathcal{D}_w in time $O(1)$ during the sampling phase. The sampling phase can be repeated any number K of times to obtain K independent samples from \mathcal{D}_w .*

We assume that the ℓ_1 -norm of w is known in the next algorithm (in this case, we can assume without loss of generality that $\|w\|_1 = 1$). If $\|w\|_1$ is unknown, a difficulty arises at step 5.a of Algorithm 7.4 to sample from the Bernoulli distribution of parameter $\frac{\|w_H\|_1}{\|w\|_1}$. This problem was addressed in [HRRS19] by using the amplitude estimation algorithm, which resulted in an approximate sampling scheme. We do not present this case here.

1. Compute a set $H \subset [N]$ of the positions of K largest entries in w by using the **top- K maximum finding** algorithm with failure probability δ .
2. Query and store the value of w_i for all $i \in H$.
3. Apply the preprocessing phase of the algorithm of Proposition 7.5.2 to construct the data structure associated with the vector $w_H = (w_i)_{i \in H}$.
4. Compute $m = \min_{i \in H} |w_i|$ and $\|w_H\|_1 = \sum_{i \in H} |w_i|$.
5. For $k = 1, \dots, K$:
 - a) Sample $b_k \in \{0, 1\}$ from the Bernoulli distribution of parameter $\|w_H\|_1$.
 - b) If $b_k = 1$, sample $i_k \sim \left(\frac{|w_i|}{\|w_H\|_1}\right)_{i \in H}$ by using the data structure built at step 3 and the sampling phase of the algorithm of Proposition 7.5.2.
 - c) If $b_k = 0$, sample $i_k \sim \left(\frac{|w_i|}{1 - \|w_H\|_1}\right)_{i \notin H}$ by preparing the state $|w_{[N] \setminus H}\rangle$ with the algorithm of Proposition 7.3.2 on input $w_{[N] \setminus H} = (w_i)_{i \notin H}$ and m , and by measuring it in the computational basis.
6. Output i_1, \dots, i_K .

Algorithm 7.4: K -fold importance sampling for unit norm input.

Theorem 7.5.3. *Consider two integers $1 \leq K \leq N$, a real $\delta \in (0, 1)$ and a probability vector $w \in \mathbb{R}^N$ (i.e. $\|w\|_1 = 1$). Then, with probability at least $1 - \delta$, the output of Algorithm 7.4 consists of K independent samples i_1, \dots, i_K from the probability distribution \mathcal{D}_w , and the algorithm performs $O(\sqrt{KN} \log(1/\delta))$ queries to w in expectation.*

Proof. We assume that the top- K maximum finding algorithm returns a correct set H at step 1, which is the case with probability at least $1 - \delta$. The K samples i_1, \dots, i_K are distributed according to \mathcal{D}_w by a simple conditional probability argument.

Similarly to the proof of Proposition 7.4.1, we have $m \leq 1/K$. Each execution of step 5.b takes time $O(1)$ by Proposition 7.5.2, and each execution of step 5.c uses $O\left(\sqrt{\frac{Nm}{1-\|w_H\|_1}}\right)$ queries in expectation by Proposition 7.3.2. Moreover, step 5.b is executed $K\|w_H\|_1$ times in expectation, and step 5.c is executed $K(1 - \|w_H\|_1)$ times in expectation. Thus, the expected number of queries used by the algorithm at step 5 is $O\left(K\|w_H\|_1 + K(1 - \|w_H\|_1)\sqrt{\frac{Nm}{1-\|w_H\|_1}}\right) \leq O(\sqrt{KN})$ by linearity of expectation. Finally, step 1 uses $O(\sqrt{KN} \log(1/\delta))$ queries and step 2 uses K queries. \square

We finally show that the above algorithms are optimal (up to a logarithmic factor) by a simple reduction from the K -Search problem. It implies that the state preparation algorithm of Theorem 7.4.3 is also optimal.

Proposition 7.5.4. *Any bounded-error quantum algorithm that can output K independent samples from the distribution \mathcal{D}_w given oracle access to any non-zero vector $w \in \mathbb{R}^N$ must perform at least $\Omega\left(\sqrt{\frac{KN}{\log K}}\right)$ quantum queries.*

Proof. The K -Search problem asks to find the positions of K preimages of 1 in an oracle $\mathcal{O} : [N] \rightarrow \{0, 1\}$. The quantum query complexity of this problem is $\Theta(\sqrt{KN})$ [KŠW07] (see also Section 10.5). By a coupon collector argument, if we sample $\Theta(K \log K)$ independent samples from the distribution \mathcal{D}_w where $w = (\mathcal{O}(1), \dots, \mathcal{O}(N)) \in \{0, 1\}^N$, then we obtain the positions of at least K different preimages of 1 with constant success probability (if such preimages exist). It implies that generating $\Omega(K \log K)$ independent samples from any distribution \mathcal{D}_w requires using at least $\Omega(\sqrt{KN})$ quantum queries to w . \square

7.6 Discussion

In this chapter, we defined the probability vector $\left(\frac{|w_1|}{\|w\|_1}, \dots, \frac{|w_N|}{\|w\|_1}\right)$ by using the ℓ_1 -norm. Our results still hold if we replace the absolute value with any function $f : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$ to define the probability vector $\left(\frac{f(w_i)}{\sum_j f(w_j)}\right)_{i \in [N]}$ (assuming the denominator is non-zero).

In particular, we can consider the ℓ_p -norm giving $\left(\frac{|w_1|^p}{\|w\|_p^p}, \dots, \frac{|w_N|^p}{\|w\|_p^p}\right)$. We also restricted ourselves to preparing states with non-negative real amplitudes. Arbitrary phase factors can be introduced by using techniques discussed in [KM01; SLSB19].

We did not address the precision errors in our analysis. In particular, it can be relevant to replace the controlled rotation gate (which requires to calculate the arcsine function) by the comparison-based circuit defined in [SLSB19] that avoids arithmetic.

Two directions for future work are to consider the case of dynamic weight updates and to close the logarithmic gap between our lower and upper bounds.

8

Applications to Stochastic Optimization

This chapter is based on the following papers:

[HRRS19] Y. Hamoudi, P. Reberntrost, A. Rosmanis, and M. Santha. “Quantum and Classical Algorithms for Approximate Submodular Function Minimization”. In: *Quantum Information & Computation* 19.15&16 (2019), pp. 1325–1349.

[RHR+21] P. Reberntrost, Y. Hamoudi, M. Ray, X. Wang, S. Yang, and M. Santha. “Quantum Algorithms for Hedging and the Learning of Ising Models”. In: *Physical Review A* 103 (2021), p. 012418.

8.1 Introduction

Importance sampling is a commonly used method in stochastic optimization, where the decisions made by an algorithm to iterate toward a solution are guided by random choices. The advantage of using a stochastic method over a deterministic one (such as stochastic gradient descent vs. vanilla gradient descent) is often to reduce the computational time, with the risk of decreasing the accuracy. Several *variance reduction* techniques have been developed to keep the accuracy under control. Among them, importance sampling is frequently used to bias an iterative process toward the data that are the most significant. This strategy can be harder to design than data-independent approaches (such as uniform sampling), but it often leads to better convergence rates [NSW16; ZZ15; ZX17]. In this chapter, we investigate the use of quantum importance sampling to speed up stochastic optimization methods.

A quite successful recent trend in quantum computing is to design fast quantum algorithms for various optimization and machine learning problems [Aar15; BWP+17; CHI+18; AG19a]. These algorithms are often *quantum-classical hybrid*, that is partly of quantum and partly of classical nature, and designed in a modular way so that the quantum part of the algorithm can be treated as a separate building block. A frequent feature of these algorithms is that they use a classical iterative method (such as gradient descent), and they only convey classical information from one iteration to the next. In contrast to the HHL algorithm [HHL09] for example, the output of these algorithm is often inherently classical. In most cases, the source of the quantum speedup lies in the use of a quantum subroutine to perform each iteration faster than it is possible classically. Examples of such subroutines include the quantum Fourier transform, amplitude amplification and estimation, and quantum importance sampling. In this chapter, we propose two new and independent applications of the latter technique to the *Hedge algorithm* and to *submodular minimization* in the quantum oracle model.

Hedge algorithm. A paradigmatic algorithm for online learning is the Hedge algorithm by Freund and Schapire [FS97]. An allocation into different strategies is chosen for multiple rounds and each round incurs corresponding losses for each strategy. The algorithm obtains a favorable guarantee for the total losses even in an adversarial situation. In Section 8.2, we present a quantum algorithm for such online learning in an oracular setting. For T time steps and N strategies, we exhibit a quantum Hedge algorithm with a run time of about $O(T^2\sqrt{N})$ for selecting individual strategies by sampling. The quantum algorithm inherits the provable learning guarantees from the classical Hedge algorithm and it exhibits a quadratic speedup in N compared to the run time $O(TN)$ of the classical algorithm.

Submodular function minimization. Submodular functions are set functions mapping every subset of some ground set of size n into the real numbers and satisfying the diminishing returns property. Submodular minimization is an important field in discrete optimization theory due to its relevance to various branches of mathematics, computer science and economics. The current fastest algorithms for submodular minimization [LSW15; ALS20] are based on a convex relaxation of submodular functions known as the *Lovász extension* [Lov82]. In Section 8.3, we present a quantum algorithm for finding an ϵ -additive error minimizer of a submodular function with range $[-1, 1]$ in time $\tilde{O}(n^{5/4}/\epsilon^{5/2})$. Prior to our work, the best classical algorithm [CLSW17] was running in time $\tilde{O}(n^{5/3}/\epsilon^2)$ and no quantum algorithm was known. Subsequently to the publication of our work, a better *classical* algorithm that runs in time $\tilde{O}(n/\epsilon^2)$ was presented in [ALS20]. Nevertheless, we believe that our approach, namely the use of quantum importance sampling in the stochastic subgradient descent method, is general enough to be of independent interest in other similar applications.

8.1.1 Related work

There is an extensive classical literature on the use of importance sampling in stochastic optimization. We refer the reader to [Nes12; RT14; SV09; NSW16; ZZ15; ZX17] for some of the main examples in stochastic descent methods. We give some applications of quantum importance sampling to stochastic optimization that we are aware of in the quantum oracle model. Li et al. [LCW19; LWCW21] and van Apeldoorn and Gilyén [AG19b] studied the problem of solving matrix games. They used a primal-dual approach based on the multiplicative weight update (MWU) method [GK95; CHW12], where the updates are obtained by quantum importance sampling of the weight vector. Arunachalam and Maity [AM20] and Izdebski and de Wolf [IW20] also used the MWU method for the problem of boosting a weak learner. Their results are adapted from the AdaBoost [FS97] and SmoothBoost [Ser03] algorithms respectively. Our work on the Hedge algorithm is another application of the MWU method. The recent line of work on quantum SDP solving [BS17; AGGW20b; BKL+19; AG19a] builds on the classical Arora-Kale framework [AK16], and it is based on the matrix MWU method and on Gibbs sampling. Finally, Zhang, Len and Li [ZLL20] gave a perturbed descent method for escaping saddle points, where the perturbation is obtained by sampling from a distribution simulated with the Schrödinger equation.

All the above-mentioned quantum optimization algorithms assume a quantum oracle access to the input data, as we do in our work. Other related results in the quantum oracle model include general convex optimization [AGGW20a; CCLW20], fast gradient computation [Jor05; GAW19; AGGW20a; CCLW20], black-box first-order convex optimization [GKNS20], and optimization problems on graphs [DHHM06; AW20; AL20].

8.2 Hedge algorithm

Consider a game with T rounds, where we have the chance to play a *mixture* of N different strategies in each round and observe the results of our choice in the next round. The setting is “online” in the sense that the results are unknown ahead of time, which also can be seen as an idealized version of sports betting or stock market trading. The Hedge algorithm adaptively changes the mixture of strategies (a probability vector) via multiplicative weight updates [FS97]. This strategy allows for losses after T rounds that are not much worse than the minimum achievable “offline” loss. Here, “offline” means that the strategy is picked in advance without any adaptation. This difference of online and minimum achievable offline loss is often called “regret”. It can be shown that the regret of the Hedge algorithm is not worse than $\sqrt{2T \log N} + \log N$ (Proposition 8.2.1). The classical complexity for this algorithm is $O(TN)$: in each round we have to perform the multiplicative update, an effort that is proportional to N .

In this section, we provide a quantum algorithm in the online hedging scenario (Algorithm 8.2). Assuming appropriate oracles for the online loss information, we exhibit a quantum speedup for the *active setting*, where in each round a single strategy is executed and incurs a certain loss. We select this strategy by quantum importance sampling. We obtain a quantum speedup in N while the overall regret remains close to that of the classical Hedge algorithm with high probability. While T and N are arbitrary, in most applications when applied in learning theory for example, T is much smaller than N , e.g. $T = O(\log N)$. In this case, the worsening in T provided by our algorithm is acceptable.

Organization of the section. We first present the classical Hedge algorithm [FS97] in Section 8.2.1 and we recall its main property in Proposition 8.2.1. Next, we describe our quantum Hedge algorithm in Section 8.2.2. The main result is given in Theorem 8.2.3.

8.2.1 Classical Hedge algorithm

We follow [FS97] for the discussion of the classical Hedge algorithm. We are given N strategies for a game that takes T rounds. Before each time $t \in [T]$, we choose an assignment of the N strategies. This assignment shall be given by a non-zero weight vector $w^{(t)} = (w_1^{(t)}, \dots, w_N^{(t)}) \in \mathbb{R}_{\geq 0}^N$, which form the probability vector $\mathcal{D}_{w^{(t)}} = \left(\frac{w_1^{(t)}}{\|w^{(t)}\|_1}, \dots, \frac{w_N^{(t)}}{\|w^{(t)}\|_1} \right)$. The initial allocation is taken to be uniform, i.e., $w^{(1)} = (1, \dots, 1)$. The algorithm considers an online learning setting, where information arrives over time and the weights are updated accordingly. Specifically, at each time $t \in [T]$, we observe the loss vector $\ell^{(t)} = (\ell_1^{(t)}, \dots, \ell_N^{(t)}) \in [0, 1]^N$. Algorithmically, we describe this as “Receive loss vector $\ell^{(t)}$ ”, which means we obtain access to the loss vector. To avoid further complexities, we assume that each loss $\ell_i^{(t)}$ takes a constant number of bits to specify. The loss at time t is given by

$$L^{(t)} := \sum_{i=1}^N \frac{w_i^{(t)}}{\|w^{(t)}\|_1} \ell_i^{(t)} = \mathbb{E}_{i \sim \mathcal{D}_{w^{(t)}}} [\ell_i^{(t)}] \in [0, 1].$$

Algorithmically, this loss is taken into account with the statement “Suffer loss $L^{(t)}$ ”, which means we add $L^{(t)}$ to the overall loss amount. A strategy to minimize losses was shown in [FS97]. Take $\beta \in (0, 1)$. The strategy is based on multiplicative updates to the weights given the incoming loss information as $w_i^{(t+1)} = w_i^{(t)} \beta^{\ell_i^{(t)}}$, which for the full path up to t is $w_i^{(t)} = \beta^{\sum_{t'=1}^{t-1} \ell_i^{(t')}}$. The original Hedge algorithm is given in Algorithm 8.1.

The accumulated loss of this algorithm over T rounds is $L_{\mathcal{H}} = \sum_{t=1}^T L^{(t)}$. On the other hand, consider the “offline loss”, $L_{\min} = \min_{i \in [N]} \sum_{t=1}^T \ell_i^{(t)}$, which gives the minimum loss achievable when choosing the same single strategy for all rounds of the game. Freund and Schapire [FS97] proved the next regret bound for the losses of the Hedge algorithm.

1. Set $w^{(1)} = (1, \dots, 1) \in \mathbb{R}^N$.
2. For $t = 1, \dots, T$:
 - a) Receive loss vector $\ell^{(t)}$.
 - b) Suffer loss $L^{(t)} = \mathbb{E}_{i \sim \mathcal{D}_{w^{(t)}}} [\ell_i^{(t)}]$.
 - c) Update the weight vector to $w^{(t+1)} = w^{(t)} \cdot \beta^{\ell^{(t)}}$.

Algorithm 8.1: Classical Hedge algorithm [FS97].

Proposition 8.2.1 ([FS97]). *For any sequence $\ell^{(1)}, \dots, \ell^{(T)} \in [0, 1]^N$ of loss vectors, the Hedge algorithm (Algorithm 8.1) with parameter $\beta = \frac{1}{1 + \sqrt{2 \log(N/T)}}$ incurs a total loss of $L_{\mathcal{H}} = \sum_{t=1}^T L^{(t)}$ that satisfies the regret bound $L_{\mathcal{H}} \leq L_{\min} + \sqrt{2T \log(N)} + \log N$, where $L_{\min} = \min_{i \in [N]} \sum_{t=1}^T \ell_i^{(t)}$. The algorithm uses $O(TN)$ operations.*

This bound is better than the naive one $L_{\mathcal{H}} \leq L_{\min} + T$. The run time is $O(TN)$ since at every step the algorithm updates N numbers and there are T steps overall. In the next section, we provide a quantum version of this algorithm with a polynomial speedup in N .

8.2.2 Quantum Hedge algorithm

We construct a simple algorithm based on “one-shot” quantum importance sampling (Proposition 7.3.2) to improve the running time of the Hedge algorithm. We aim at a sublinear dependence in the number N of strategies, which means that we cannot update and store the entire weight vector $w^{(t)}$. We resolve this problem by considering the *active setting* where in each round the algorithm explicitly chooses a single strategy to play. In comparison, the classical Hedge algorithm presented in Section 8.2.1 is formulated in the *passive setting*, where all N strategies could be played at the same time by investing a proportion $\frac{w_i^{(t)}}{\|w^{(t)}\|_1}$ of the available resources in the i -th strategy (think for instance of allocating a portfolio). We show that playing a single strategy picked according to the distribution $\mathcal{D}_{w^{(t)}}$ does not significantly increase the regret bound compared to the passive setting with high probability. We note that a similar approach can be implemented with a randomized algorithm, but the run time would still be $O(TN)$ since sampling from the distribution $\mathcal{D}_{w^{(t)}}$ requires N operations classically.

Before describing the quantum Hedge algorithm, we translate the online learning setting into the quantum domain. The input data for the quantum algorithm are the losses experienced at every step t . First, we assume T different oracles, where the sequential access to these oracles embodies the online setting.

Assumption 8.A (LOSS ORACLES). Assume that $s = O(1)$ bits are sufficient to specify the losses $\ell_i^{(t)}$ and the weights $w_i^{(t)}$ obtained in the Hedge algorithm. For $t \in [T]$ and $i \in [N]$, assume query unitaries $\mathcal{O}_{\ell^{(t)}}$ such that $\mathcal{O}_{\ell^{(t)}}(|i\rangle|v\rangle) = |i\rangle|v \oplus \ell_i^{(t)}\rangle$ for any $v \in \{0, 1\}^s$.

Given the loss unitaries, we can compute the weights with overhead about $O(T)$.

Fact 8.2.2. *Let $t \in [T]$, $\beta \in (0, 1)$ and $\mathcal{O}_{\ell^{(t)}}$ for $t' \in [t-1]$ be given as in Assumption 8.A. Then, there is a unitary circuit \mathcal{C} such that $\mathcal{C}(|i\rangle|v\rangle) = |i\rangle|v \oplus w_i^{(t)}\rangle$ where $w_i^{(t)} = \beta^{\sum_{t'=1}^{t-1} \ell_i^{(t')}}$. This computation takes $O(t)$ queries to the data input and $O(t + \log N)$ other gates.*

We use these unitaries to perform quantum importance sampling in the quantum Hedge algorithm given below. The algorithm never fully exhibit the full weight vector but rather only its coherent encoding $|w^{(t)}\rangle$ over $O(\log N)$ qubits, hence the sublinear running time.

1. Let $w^{(1)} = (1, \dots, 1) \in \mathbb{R}^N$. Set up a circuit $\mathcal{C}^{(1)}$ that gives query access to $w^{(1)}$.
2. For $t = 1, \dots, T$:
 - a) Compute $m = \|w^{(t)}\|_\infty$ with failure probability $\delta/(2T)$ by using the maximum finding algorithm (Proposition 7.3.1) and the query access to $w^{(t)}$ given by $\mathcal{C}^{(t)}$.
 - b) Sample $i^{(t)} \sim \mathcal{D}_{w^{(t)}}$ by using the importance sampling algorithm (Proposition 7.3.2) with m and the query access to $w^{(t)}$ given by $\mathcal{C}^{(t)}$.
 - c) Receive access to the loss vector $\ell^{(t)}$ via the unitary $\mathcal{O}_{\ell^{(t)}}$ of Assumption 8.A.
 - d) Suffer loss $\ell_{i^{(t)}}^{(t)}$.
 - e) Use Fact 8.2.2 to set up a circuit $\mathcal{C}^{(t+1)}$ that gives query access to $w^{(t+1)} = w^{(t)} \cdot \beta^{\ell^{(t)}}$.

Algorithm 8.2: Quantum Hedge algorithm.

Theorem 8.2.3. *For any sequence $\ell^{(1)}, \dots, \ell^{(T)} \in [0, 1]^N$ of loss vectors and any real $\delta \in (0, 1)$, the quantum Hedge Algorithm (Algorithm 8.2) with parameter $\beta = \frac{1}{1 + \sqrt{2 \log(N/T)}}$ incurs a total loss of $L_{\mathcal{H}} = \sum_{t=1}^T \ell_{i^{(t)}}^{(t)}$ that satisfies the regret bound*

$$L_{\mathcal{H}} \leq L_{\min} + 2\sqrt{T \log(N/\delta)} + \log N$$

with probability at least $1 - \delta$, where $L_{\min} = \min_{i \in [N]} \sum_{t=1}^T \ell_i^{(t)}$. It uses $O(T^2 \sqrt{N} \log(T/\delta))$ quantum queries to the loss vectors and $\tilde{O}(T^2 \sqrt{N} \log(1/\delta))$ other gates.

Proof. Let \mathcal{E} denote the event that all calls to the maximum finding algorithm at step 2.a are successful. We have $\Pr[\mathcal{E}] \geq \delta/2$ by a union bound. Conditioned on \mathcal{E} , the expected loss suffered by the algorithm at the t -th iteration is equal to $\mathbb{E}_{i \sim \mathcal{D}_{w^{(t)}}}[\ell_i^{(t)}]$. Thus, the loss $L_{\mathcal{H}}$ satisfies that $\mathbb{E}[L_{\mathcal{H}} | \mathcal{E}] = \sum_{t=1}^T \mathbb{E}_{i \sim \mathcal{D}_{w^{(t)}}}[\ell_i^{(t)}]$. By Proposition 8.2.1, we have

$$\mathbb{E}[L_{\mathcal{H}} | \mathcal{E}] \leq L_{\min} + \sqrt{2T \log(N)} + \log N.$$

The loss $L_{\mathcal{H}}$ is a sum of T random variables distributed in $[0, 1]$. Thus, by Hoeffding's inequality, $\Pr[L_{\mathcal{H}} \geq \mathbb{E}[L_{\mathcal{H}} | \mathcal{E}] + \sqrt{T \log(2/\delta)/2} | \mathcal{E}] \leq \delta/2$. We conclude that,

$$L_{\mathcal{H}} \leq L_{\min} + \sqrt{2T \log(N)} + \log N + \sqrt{T \log(2/\delta)/2} \leq L_{\min} + 2\sqrt{T \log(N/\delta)} + \log N$$

with probability at least $(1 - \delta/2)^2$. Each circuit $\mathcal{C}^{(t)}$ uses $O(t)$ queries to $\ell^{(1)}, \dots, \ell^{(t-1)}$ to answer one query to $w^{(t)}$ by Fact 8.2.2. Consequently, at the t -th iteration, the algorithm uses $O(t \cdot \sqrt{N} \log(1/\delta))$ queries in step 2.a by Proposition 7.3.1 and $O(t \cdot \sqrt{N})$ queries in step 2.b by Proposition 7.3.2. The total number of queries is $O(T^2 \sqrt{N} \log(1/\delta))$. \square

8.3 Submodular function minimization

A submodular function F is a function mapping every subset of some finite set V of size n into the real numbers and satisfying the diminishing returns property: for every $A \subseteq B \subseteq V$ and for every $i \notin B$, the inequality $F(A \cup \{i\}) - F(A) \geq F(B \cup \{i\}) - F(B)$ holds. In other words, given two sets where one of them contains the other, adding a new item to the smaller set increases the function value at least as much as adding that element to the bigger set. Many classical functions in mathematics, computer science and economics are submodular, the most prominent examples include entropy functions, cut capacity functions, matroid rank functions and utility functions. Submodular functions show analogies both with concavity and convexity. The diminishing returns property makes them akin to concave functions, but they have algorithmic properties similar to convex functions. In particular, while it follows from the NP-hardness of maximum cut that submodular maximization is NP-hard, submodular minimization can be solved in polynomial time, in fact even in strongly polynomial time.

The link between submodular functions and convex analysis is made explicit through the *Lovász extension* [Lov82]. There are various approaches to solve submodular minimization. The foundational work of Grötschel, Lovász and Schrijver [GLS81] gave the first polynomial-time algorithm using the ellipsoid method. The first pseudo-polynomial algorithm using a combinatorial method appeared in the influential paper of Cunningham [Cun85]. In a later work Grötschel, Lovász and Schrijver [GLS88] were the first to design a strongly polynomial-time algorithm, and the first strongly polynomial-time combinatorial algorithms were given by Schrijver [Sch00] and by Iwata, Fleischer and Fujishige [IFF01]. The current fastest submodular minimization algorithm is by Lee, Sidford and Wong [LSW15]. Many of these works assume an access to a query oracle for the function F .

Our work is most closely related to the paper of Chakrabarty et al. [CLSW17] who gave an ϵ -additive approximation algorithm that runs in time $\tilde{O}(n^{5/3}/\epsilon^2)$ for real-valued submodular functions with range $[-1, 1]$. This algorithm was the first to run in subquadratic time in n . In [HRRS19], we improved this result by giving a classical algorithm that runs in time $\tilde{O}(n^{3/2}/\epsilon^2)$, and a quantum algorithm that runs in time $\tilde{O}(n^{5/4}/\epsilon^{5/2})$. We present a slightly simpler version of the latter result in this section. Our method consists of minimizing the Lovász extension of the submodular function under consideration by using the stochastic subgradient descent algorithm. We differ from [CLSW17] by constructing a new subgradient estimator that is faster to evaluate with quantum importance sampling. Axelrod, Liu and Sidford [ALS20] have published subsequently to our work a classical nearly linear time algorithm that outperforms our result.

Theorem 8.3.17 (Restated). *There exists a quantum algorithm such that, given a submodular function $F : 2^V \rightarrow [-1, 1]$ and a real $\epsilon \in (\frac{1}{n^{1/6}}, 1)$, it computes a set \bar{S} such that $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ in time $\tilde{O}(n^{5/4}/\epsilon^{5/2})$.*

Organization of the section. We first present a high overview of the algorithm in Section 8.3.1. Next, we introduce the basic definitions and properties of submodular functions in Section 8.3.2. We analyze in Proposition 8.3.5 a variant of the stochastic subgradient descent method, where the subgradient estimates need not be unbiased. We describe the data structures that are needed for our algorithm in Section 8.3.3. The quantum importance sampling algorithm developed in the previous chapter is used in Section 8.3.4 to construct two types of subgradient estimates. The final algorithm is presented in Section 8.3.5.

8.3.1 Proof overview

Submodular minimization can be translated into a convex optimization problem by considering the so-called *Lovász extension* f . This makes it possible to apply standard gradient algorithms. Since f is not differentiable, one can rely on the subgradient descent method that computes a sequence of iterates $x^{(t)}$ converging to a minimum of f . At each step, the next iterate $x^{(t+1)}$ is obtained by moving into the negative direction of a subgradient $g^{(t)}$ at $x^{(t)}$. In the case of submodular functions, there exists a natural choice for $g^{(t)}$, sometimes called the *Lovász subgradient*, that requires $O(n/\epsilon^2)$ steps to converge to an ϵ -approximate of the minimum. The *stochastic subgradient descent method* allows one to replace the subgradient $g^{(t)}$ with a stochastic subgradient, that is a low-variance estimate $\tilde{g}^{(t)}$ satisfying $\mathbb{E}[\tilde{g}^{(t)} | x^{(t)}] = g^{(t)}$. All the existing works on approximate submodular minimization [HK12; Bac13; CLSW17; HRRS19; ALS20], including the present one, rely on computing such an estimate in the most efficient way.

We now explain how we compute a stochastic subgradient $\tilde{g}^{(t)}$. We simplify the exposition by assuming that ϵ is a small constant. One natural choice for $\tilde{g}^{(t)}$ is the *subgradient direct estimate* $\hat{g}^{(t)} = \|g^{(t)}\|_1 \text{sgn}(g_i^{(t)}) \cdot e_i$ where $i \in [n]$ is chosen by sampling from the distribution $\mathcal{D}_{g^{(t)}}$ that gives i with probability $|g_i^{(t)}|/\|g^{(t)}\|_1$, and e_i is the basis vector with a 1 at position i . The ℓ_1 -norm of the Lovász subgradient being small, this is a valid low-variance 1-sparse estimate of $g^{(t)}$. The quantum importance sampling algorithm can be used to compute $\hat{g}^{(t)}$ in time $\tilde{O}(\sqrt{n})$, which already leads to an algorithm with overall complexity $\tilde{O}(n^{3/2})$. We improve upon this complexity by constructing $\tilde{g}^{(t)}$ differently, in amortized time $\tilde{O}(n^{1/4})$. Similarly to [CLSW17], we construct our subgradient estimate $\tilde{g}^{(t)}$ by combining two kinds of estimates (Proposition 8.3.14). Our construction is reset every $K = \sqrt{n}$ steps of the descent, which turns out to be the optimal resetting time in our case. We explain how to compute the first K terms $\tilde{g}^{(0)}, \dots, \tilde{g}^{(K-1)}$. First, we obtain K independent samples $\hat{g}^{(0,0)}, \dots, \hat{g}^{(0,K-1)}$ from the subgradient direct estimate at $x^{(0)}$ by using the K -fold importance sampling algorithm (Theorem 7.5.1) in time $\tilde{O}(\sqrt{nK})$ (Proposition 8.3.12). Then, the first subgradient estimate is chosen to be $\tilde{g}^{(0)} = \hat{g}^{(0,0)}$, and the other ones are obtained at step t by combining $\hat{g}^{(0,t)}$ with an estimate $\tilde{d}^{(t)}$ of the Lovász subgradient difference $d^{(t)} = g^{(t)} - g^{(0)}$, that is $\tilde{g}^{(t)} = \hat{g}^{(0,t)} + \tilde{d}^{(t)}$. Our procedure for constructing $\tilde{d}^{(t)}$ (Proposition 8.3.13) is adapted from [CLSW17] and it uses the “one-shot” quantum importance sampling algorithm (Proposition 7.3.2) to run in time $\tilde{O}(\sqrt{t})$. Therefore, the first K estimates are obtained in time $\tilde{O}(\sqrt{nK} + \sum_{t=1}^{K-1} \sqrt{t}) = \tilde{O}(n^{3/4})$. Since the $O(n)$ steps of the subgradient descent are split into $O(n/K)$ batches of length K , it follows that the total time complexity is $\tilde{O}(n^{5/4})$.

8.3.2 Preliminaries

A submodular function is a set function $F : 2^V \rightarrow \mathbb{R}$ such that for every $A \subseteq B \subseteq V$ and for every $i \notin B$, the inequality $F(A \cup \{i\}) - F(A) \geq F(B \cup \{i\}) - F(B)$ holds. For convenience, we assume that $V = [n]$ and $F(\emptyset) = 0$ (this can be enforced by observing that $S \mapsto F(S) - F(\emptyset)$ is still a submodular function). The Lovász extension $f : [0, 1]^n \rightarrow \mathbb{R}$ is a convex relaxation of F to the hypercube $[0, 1]^n$. Before describing it, we present a canonical way to associate a permutation P with each point $x \in [0, 1]^n$.

Definition 8.3.1. Given a permutation $P = (P_1, \dots, P_n)$ of $[n]$, we say that P is *consistent* with $x \in \mathbb{R}^n$ if $x_{P_1} \geq x_{P_2} \geq \dots \geq x_{P_n}$, and $P_{i+1} > P_i$ when $x_{P_i} = x_{P_{i+1}}$ for all i . We also let $P[i] = \{P_1, \dots, P_i\} \subseteq [n]$ denote the set of the first i elements of P , and $P[0] = \emptyset$.

For instance, the permutation P consistent with $x = (0.3, 0.2, 0.3, 0.1)$ is $P = (1, 3, 2, 4)$.

Definition 8.3.2 (LOVÁSZ EXTENSION AND LOVÁSZ SUBGRADIENT). Given a submodular function $F : 2^V \rightarrow \mathbb{R}$ over $V = [n]$, the *Lovász extension* $f : [0, 1]^n \rightarrow \mathbb{R}$ of F is defined for all $x \in [0, 1]^n$ by $f(x) = \sum_{i \in [n]} (F(P[i]) - F(P[i-1])) \cdot x_{P_i}$ where P is the permutation consistent with x . The *Lovász subgradient* $g(x) \in \mathbb{R}^n$ at $x \in [0, 1]^n$ is defined by $g(x)_{P_i} = F(P[i]) - F(P[i-1])$ for all $i \in [n]$.

We will use the following properties of the Lovász extension [Lov82; Bac13; JB11].

Proposition 8.3.3. *The Lovász extension f of a submodular function F is a convex function. Moreover, given $x \in [0, 1]^n$ and the permutation P consistent with x , we have*

- (1) (Subgradient) *For all $y \in [0, 1]^n$, $\langle g(x), x - y \rangle \geq f(x) - f(y)$.*
- (2) (Minimizers) $\min_{i \in [n]} F(P[i]) \leq f(x)$ and $\min_{S \subseteq V} F(S) = \min_{y \in [0, 1]^n} f(y)$.
- (3) (Boundedness) *If the range of F is $[-1, 1]$ then $\|g(x)\|_2 \leq \|g(x)\|_1 \leq 3$.*

Observe that the second property gives an explicit way to convert any point $\bar{x} \in [0, 1]^n$ such that $f(\bar{x}) \leq \min_{x \in [0, 1]^n} f(x) + \epsilon$ into a set $\bar{S} \subseteq V$ such that $F(\bar{S}) \leq \min_{S \subseteq V} F(S) + \epsilon$. Consequently, we can focus on ϵ -additive minimization of the Lovász extension.

Model of computation. Although the Lovász extension f is a continuous function, the value of $f(x)$ can be computed by evaluating F on the sets $P[1], \dots, P[n]$ where P is the permutation consistent with x . Consequently, it is natural to define a query oracle that given i returns $F(P[i])$ [CLSW17]. More precisely, given a permutation P stored in a linked list, we assume that we have access to a unitary \mathcal{O}_P such that $\mathcal{O}_P(|i\rangle|0\rangle) = |i\rangle|F(P[i])\rangle$ for any $i \in [n]$, where the second register holds a binary representation of $F(P[i])$ with some finite precision. The Lovász extension can be evaluated with $O(n)$ queries and $O(n \log n)$ other operations. The time complexity is measured as the number of uses of \mathcal{O}_P .

Stochastic subgradient descent. The stochastic subgradient descent method is a general algorithm for approximating the minimum value of a convex function f that is not necessarily differentiable (as it is the case for the Lovász extension). It uses the concept of *subgradients* (or *subderivatives*) of f , which is defined as follows.

Definition 8.3.4. Given a convex function $f : C \rightarrow \mathbb{R}$ over a set $C \subseteq \mathbb{R}^n$ and a point $x \in C$, we say that $g \in \mathbb{R}^n$ is a *subgradient* of f at x if $\langle g, x - y \rangle \geq f(x) - f(y)$ for all $y \in C$. The set of all subgradients at x is denoted by $\partial f(x)$.

The stochastic subgradient descent method requires computing a sequence $(\tilde{g}^{(t)})_t$ of *unbiased* subgradient estimates at certain points $(x^{(t)})_t$, that is $\mathbb{E}[\tilde{g}^{(t)} | x^{(t)}] \in \partial f(x^{(t)})$. We generalize this method to ϵ -noisy estimates such that $\|\mathbb{E}[\tilde{g}^{(t)} | x^{(t)}] - g^{(t)}\|_1 \leq \epsilon$ for some $g^{(t)} \in \partial f(x^{(t)})$. In the case $\epsilon = 0$, our analysis recovers the standard error bound [Duc18].

Proposition 8.3.5 (NOISY STOCHASTIC SUBGRADIENT DESCENT). *Let $f : C \rightarrow \mathbb{R}$ be a convex function over a compact convex set $C \subset \mathbb{R}^n$, and fix a step size parameter $\eta > 0$. Consider two sequences of random variables $(x^{(t)})_t$ and $(\tilde{g}^{(t)})_t$ such that $x^{(0)} = \operatorname{argmin}_{x \in C} \|x\|_2$, $x^{(t+1)} = \operatorname{argmin}_{x \in C} \|x - (x^{(t)} - \eta \tilde{g}^{(t)})\|_2$, and*

$$\|\mathbb{E}[\tilde{g}^{(t)} | x^{(t)}] - g^{(t)}\|_1 \leq \epsilon \text{ for some } g^{(t)} \in \partial f(x^{(t)}),$$

for all $t \geq 0$. Fix $x^ \in \operatorname{argmin}_{x \in C} f(x)$ and let $L_2, L_\infty, B \in \mathbb{R}$ be such that $\|x - x^*\|_2 \leq L_2$, $\|x - x^*\|_\infty \leq L_\infty$ and $\mathbb{E}[\|\tilde{g}^{(t)}\|_2^2] \leq B^2$, for all $x \in C$ and $t \geq 0$. Then, for any integer T , the average point $\bar{x} = \frac{1}{T} \sum_{t=0}^{T-1} x^{(t)}$ satisfies $\mathbb{E}[f(\bar{x})] \leq f(x^*) + \frac{L_2^2}{2\eta T} + \frac{\eta}{2} B^2 + \epsilon L_\infty$.*

Proof. Let $(g^{(t)})_t$ be such that $g^{(t)} \in \partial f(x^{(t)})$ and $\|\mathbb{E}[\tilde{g}^{(t)} | x^{(t)}] - g^{(t)}\|_1 \leq \epsilon$. Then,

$$\begin{aligned} \|x^{(t+1)} - x^*\|_2^2 &= \left\| \underset{x \in C}{\operatorname{argmin}} \|x - (x^{(t)} - \eta \tilde{g}^{(t)})\|_2 - x^* \right\|_2^2 \\ &\leq \|x^{(t)} - \eta \tilde{g}^{(t)} - x^*\|_2^2 \quad \text{by property of the projection onto } C \\ &= \|x^{(t)} - x^*\|_2^2 - 2\eta \langle \tilde{g}^{(t)}, x^{(t)} - x^* \rangle + \eta^2 \|\tilde{g}^{(t)}\|_2^2 \\ &= \|x^{(t)} - x^*\|_2^2 - 2\eta \langle g^{(t)}, x^{(t)} - x^* \rangle - 2\eta \langle \tilde{g}^{(t)} - g^{(t)}, x^{(t)} - x^* \rangle + \eta^2 \|\tilde{g}^{(t)}\|_2^2 \\ &\leq \|x^{(t)} - x^*\|_2^2 - 2\eta(f(x^{(t)}) - f(x^*)) - 2\eta \langle \tilde{g}^{(t)} - g^{(t)}, x^{(t)} - x^* \rangle + \eta^2 \|\tilde{g}^{(t)}\|_2^2 \end{aligned}$$

where the last line is by definition of a subgradient. We now take the expectation of the above formula. By the law of total expectation, we have $\mathbb{E}[\langle \tilde{g}^{(t)} - g^{(t)}, x^{(t)} - x^* \rangle] = \mathbb{E}[\langle \mathbb{E}[\tilde{g}^{(t)} | x^{(t)}] - g^{(t)}, x^{(t)} - x^* \rangle]$ and by Hölder's inequality $|\langle \mathbb{E}[\tilde{g}^{(t)} | x^{(t)}] - g^{(t)}, x^{(t)} - x^* \rangle| \leq \|\mathbb{E}[\tilde{g}^{(t)} | x^{(t)}] - g^{(t)}\|_1 \cdot \|x^{(t)} - x^*\|_\infty \leq \epsilon L_\infty$. Consequently,

$$\mathbb{E}[\|x^{(t+1)} - x^*\|_2^2] - \mathbb{E}[\|x^{(t)} - x^*\|_2^2] \leq -2\eta \mathbb{E}[f(x^{(t)}) - f(x^*)] + 2\eta \epsilon L_\infty + \eta^2 B^2.$$

By reordering this formula, we obtain that $\mathbb{E}[f(x^{(t)})] \leq f(x^*) + \frac{1}{2\eta}(\mathbb{E}[\|x^{(t)} - x^*\|_2^2] - \mathbb{E}[\|x^{(t+1)} - x^*\|_2^2]) + \frac{\eta}{2}B^2 + \epsilon L_\infty$. Finally, we upper bound the expected value of the function at the average point \bar{x} as

$$\begin{aligned} \mathbb{E}[f(\bar{x})] &\leq \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[f(x^{(t)})] \quad \text{by convexity} \\ &\leq f(x^*) + \frac{1}{T} \sum_{t=0}^{T-1} \frac{1}{2\eta} \left(\mathbb{E}[\|x^{(t)} - x^*\|_2^2] - \mathbb{E}[\|x^{(t+1)} - x^*\|_2^2] \right) + \frac{\eta}{2}B^2 + \epsilon L_\infty \\ &= f(x^*) + \frac{1}{2\eta T} \left(\mathbb{E}[\|x^{(0)} - x^*\|_2^2] - \mathbb{E}[\|x^{(T)} - x^*\|_2^2] \right) + \frac{\eta}{2}B^2 + \epsilon L_\infty \\ &\leq f(x^*) + \frac{L_2^2}{2\eta T} + \frac{\eta}{2}B^2 + \epsilon L_\infty \end{aligned}$$

where we have used the telescoping property of the sum in the third line. \square

8.3.3 Data structures and c -covers

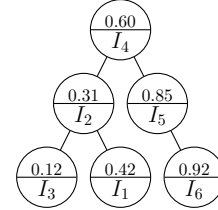
In the rest of this chapter, f denotes the Lovász extension and g denotes the Lovász subgradient (Definition 8.3.2). We describe two data structures D_1 and D_2 that will be maintained throughout the algorithm. These data structures are used in Section 8.3.4 to construct the subgradient estimates needed in the stochastic subgradient descent method. The first data structure $D_1(x)$ provides a fast access to the permutation P consistent with a point x .

Definition 8.3.6 (DATA STRUCTURE D_1). Given $x \in \mathbb{R}^n$ and the permutation P consistent with x , we define $D_1(x) = (L_x, A_x, T_x)$ to be the data structure made of the following elements: a doubly linked list L_x storing P , an array A_x storing at position $i \in [n]$ the value x_i with a pointer to the corresponding entry in P , and a self-balancing binary search tree T_x (e.g. a red-black tree [GS78]) with a node for each $i \in [n]$ keyed by the value x_i and containing the size of its subtree.

The second data structure $D_2(x, y, \mathcal{I})$ requires the following definition of a c -cover introduced in [CLSW17].

Definition 8.3.7 (*c*-COVER). Consider $x, y \in [0, 1]^n$ and let P and Q be the permutations consistent with x and y respectively. We say that a partition $\mathcal{I} = \{I_1, \dots, I_c\}$ of $[n]$ is a *c-cover* of (x, y) if, for each $j \in [c]$, the preimage of I_j under both P and Q is a set of consecutive numbers, and $x_i = y_i$ for all $i \in I_j$ if $|I_j| > 1$.

	1	2	3	4	5	6	7	8	9	10
x	0.85	0.58	0.42	0.53	0.60	0.78	0.12	0.27	0.92	0.31
y	0.85	0.58	0.65	0.53	0.60	0.78	0.90	0.27	0.92	0.31
P	9	1	6	5	2	4	(3)	10	8	(7)
Q	9	(7)	1	6	(3)	5	2	4	10	8



$$I_1 = \{3\}, I_2 = \{10, 8\}, I_3 = \{7\}, I_4 = \{5, 2, 4\}, I_5 = \{1, 6\}, I_6 = \{9\}.$$

Figure 8.3: An illustration of a 6-cover $\mathcal{I} = \{I_1, \dots, I_6\}$ for some $x, y \in [0, 1]^{10}$ and their corresponding permutations P, Q . The circled numbers in the array correspond to the positions where x and y differ (these values must belong to singletons in the cover). The binary tree corresponds to $T_x^{\mathcal{I}}$ in $D_2(x, y, \mathcal{I})$.

An example of a 6-cover is given in Figure 8.3. Note that there exists a cover of size at most $3c + 1$ if the difference vector $z = x - y$ is *c*-sparse. Later on, we will store a cover approaching that size. We describe the data structure $D_2(x, y, \mathcal{I})$ used to store a *c*-cover.

Definition 8.3.8 (DATA STRUCTURE D_2). Given $x, y \in \mathbb{R}^n$ and a *c*-cover $\mathcal{I} = \{I_1, \dots, I_c\}$ of (x, y) , we define $D_2(x, y, \mathcal{I}) = (D_1(x), D_1(y), A_x^{\mathcal{I}}, A_y^{\mathcal{I}}, T_x^{\mathcal{I}}, T_y^{\mathcal{I}})$ to be the data structure made of the following elements: $D_1(x)$ and $D_1(y)$ (Definition 8.3.6), two dynamic arrays $A_x^{\mathcal{I}}$ and $A_y^{\mathcal{I}}$ of size c storing at position $j \in [c]$ the pairs $(\arg\max_{i \in I_j} x_i, \arg\min_{i \in I_j} x_i)$ and $(\arg\max_{i \in I_j} y_i, \arg\min_{i \in I_j} y_i)$ respectively, two self-balancing binary search trees $T_x^{\mathcal{I}}$ and $T_y^{\mathcal{I}}$ with a node for each $j \in [c]$ keyed by the value of $\max_{i \in I_j} x_i$ and $\max_{i \in I_j} y_i$.

The next lemma is the crucial property established in [CLSW17] about *c*-covers. It shows that the coordinates $g(y)_i - g(x)_i$ of the Lovász subgradient difference have constant sign over any set I_j of the cover when $x \leq y$ or $x \geq y$. In particular, the ℓ_1 -norm $\|g(y)_{I_j} - g(x)_{I_j}\|_1$ is equal to the absolute value of $\sum_{i \in I_j} g(y)_i - g(x)_i$.

Lemma 8.3.9 ([CLSW17]). Consider $x, y \in [0, 1]^n$ such that $x \leq y$ or $x \geq y$, and let $\{I_1, \dots, I_c\}$ be a *c*-cover of (x, y) . Then, for each $j \in [c]$, the coordinates $g(y)_i - g(x)_i$ have the same sign for all $i \in I_j$. In particular, $|\sum_{i \in I_j} g(y)_i - g(x)_i| = \|g(y)_{I_j} - g(x)_{I_j}\|_1$.

Proof. Let P and Q denote the permutations consistent with x and y respectively. Consider $j \in [c]$ such that $|I_j| > 1$ (the result is trivial when $|I_j| = 1$). By definition of a *c*-cover, there exist three integers a_j, a'_j, ℓ_j such that $I_j = \{P_{a_j}, P_{a_j+1}, \dots, P_{a_j+\ell_j}\} = \{Q_{a'_j}, Q_{a'_j+1}, \dots, Q_{a'_j+\ell_j}\}$. Assume that $x \leq y$ (the case $x \geq y$ is symmetric). Since $x_i = y_i$ for all $i \in I_j$, we must have $P[a_j - 1] \subseteq Q[a'_j - 1]$. Thus, by the diminishing returns property of submodular functions, $F(P[a_j + \ell] - F(P[a_j + \ell - 1]) \geq F(Q[a'_j + \ell] - F(Q[a'_j + \ell - 1]))$ for all $0 \leq \ell \leq \ell_j$. We conclude that $g(y)_i - g(x)_i \leq 0$ for all $i \in I_j$. \square

Note that the condition $x \leq y$ or $x \geq y$ is crucial in the above result, which will require us to decompose a vector into a positive and a negative part in the final algorithm (step 3.c of Algorithm 8.6). We now describe three useful operations that can be handled in logarithmic time by using $D_2(x, y, \mathcal{I})$ and the above lemma. The first two operations originate from the work of [CLSW17], whereas the third one is new to this work.

Proposition 8.3.10. *Let $x, y \in \mathbb{R}^n$ such that $x \leq y$ or $x \geq y$, and let $\mathcal{I} = \{I_1, \dots, I_c\}$ be a c -cover of (x, y) . Then, given the data structure $\mathcal{D}_2(x, y, \mathcal{I})$, the following operations can be handled with $O(1)$, $O(\log n)$ and 0 queries respectively, and $O(\log n)$ other operations.*

- (1) (Subnorm) *Given $j \in [c]$, output $\|g(y)_{I_j} - g(x)_{I_j}\|_1$. This operation can also be performed coherently (meaning that the mapping $|j\rangle|0\rangle \mapsto |j\rangle\|g(y)_{I_j} - g(x)_{I_j}\|_1\rangle$ can be implemented with some garbage qubits restored to 0).*
- (2) (Subsampling) *Given $j \in [c]$, sample $i \sim \mathcal{D}_{g(y)_{I_j} - g(x)_{I_j}}$.*
- (3) (Update) *Given a 1-sparse vector $z \in \mathbb{R}^n$, update the data structure to $\mathcal{D}_2(x + z, y, \mathcal{I}')$ where \mathcal{I}' is a cover of $(x + z, y)$ of size at most $c + 3$.*

Proof. Let P be the permutation consistent with x . Observe that the rank P_i^{-1} of any coordinate x_i can be computed in $O(\log n)$ time by using T_x (since each node in the tree contains the size of its subtree).

(Subnorm) By definition of a c -cover, there exist $a_j \leq b_j$ such that $I_j = \{P_{a_j}, P_{a_j+1}, \dots, P_{b_j}\}$. Thus, $\sum_{i \in I_j} g(x)_i = \sum_{i=a_j}^{b_j} F(P[i]) - F(P[i-1]) = F(P[b_j]) - F(P[a_j-1])$. Since a_j and b_j can be obtained in time $O(\log n)$ by using $\mathcal{D}_2(x, y, \mathcal{I})$, this sum can be computed with 1 query and $O(\log n)$ other operations, and similarly for $\sum_{i \in I_j} g(y)_i$. According to Lemma 8.3.9, the difference $|\sum_{i \in I_j} g(y)_i - g(x)_i|$ is equal to the ℓ_1 -norm of $g(y)_{I_j} - g(x)_{I_j}$.

(Subsampling) We find the highest node $i_h \in I_j$ in the tree T_x , and we compute its rank $r_j = P_{i_h}^{-1}$ in time $O(\log n)$. We partition I_j into $A = \{P_{a_j}, P_{a_j+1}, \dots, P_{r_j-1}\}$, $B = \{P_{r_j}\}$, $C = \{P_{r_j+1}, \dots, P_{b_j}\}$ and we compute the ℓ_1 -norm of $g(y) - g(x)$ restricted to each set with $O(1)$ queries. We select A , B or C with probability $\frac{\|g(y)_A - g(x)_A\|_1}{\|g(y)_{I_j} - g(x)_{I_j}\|_1}$, $\frac{\|g(y)_B - g(x)_B\|_1}{\|g(y)_{I_j} - g(x)_{I_j}\|_1}$ and $\frac{\|g(y)_C - g(x)_C\|_1}{\|g(y)_{I_j} - g(x)_{I_j}\|_1}$ respectively. If we obtain a singleton, we terminate and output the value it contains, otherwise we sample recursively in the corresponding subtree of T_x .

(Update) We detail the update of the c -cover (the other parts being standard to update). Let i be the position where $z_i \neq 0$ (assuming $z \neq 0$), and denote $r = P_i^{-1}$ its rank in P before the update. First, split the set $I_j = \{P_{a_j}, \dots, P_{r-1}, i, P_{r+1}, \dots, P_{b_j}\}$ containing i into three parts $\{P_{a_j}, P_{a_j+1}, P_{r-1}\}$, $\{i\}$ and $\{P_{r+1}, \dots, P_{b_j}\}$. Then, identify the rank r' such that $x_{P_{r'}} > x_i + z_i > x_{P_{r'+1}}$ and split the set containing $P_{r'}$ into two parts. These operations can be done in $O(\log n)$ time. The size of the cover is increased by at most 3. \square

8.3.4 Importance sampling for gradient computation

We describe two algorithms for computing a noisy Lovász subgradient and a noisy Lovász subgradient difference by using the quantum importance sampling algorithm developed in Chapter 7 and the data structures described in the previous section. We need the next folklore result for estimating the norm of an n -dimensional vector given query access to it.

Lemma 8.3.11 (NORM ESTIMATION). *There is a quantum algorithm such that, given a query oracle to a non-zero vector $w \in \mathbb{R}^n$, the value $m = \|w\|_\infty$ and two reals $0 < \epsilon, \delta < 1$, it outputs a norm estimate $\tilde{\gamma}$ such that $|\tilde{\gamma} - \|w\|_1| \leq \epsilon \|w\|_1$ with probability at least $1 - \delta$. The query complexity of this algorithm is $O((\sqrt{n}/\epsilon) \log(1/\delta))$.*

Proof. Consider the real-valued random variable X that takes value $n|w_i|$ with probability $1/n$ for each $i \in [n]$. We have $\mathbb{E}[X] = \|w\|_1 \geq m$ and $0 \leq X \leq nm$. We use the Bernoulli estimator $\text{BernEst}(X, t, 0, b, \delta)$ (Proposition 4.4.1) with $t = \frac{2\sqrt{n}}{\epsilon} \log(1/\delta)$ and $b = nm$ to obtain an estimate $\tilde{\gamma}$ such that $|\tilde{\gamma} - \|w\|_1| \leq \frac{\sqrt{b\|w\|_1 \log(1/\delta)}}{t} + \frac{b \log(1/\delta)^2}{t^2} \leq \epsilon \|w\|_1$ with probability at least $1 - \delta$ in time $O(t)$. \square

Our first result is a gradient sampling algorithm **GSample** that produces a batch of K estimates of the Lovász subgradient $g(x)$ at any x . This is a simple but expensive procedure that does not use the properties of the Lovász subgradient, apart from its boundedness.

1. Compute the maximum value $m = \|g(x)\|_\infty$ with failure probability $\delta/3$ by using the maximum finding algorithm (Proposition 7.3.1).
2. Compute an estimate $\tilde{\gamma}$ of the norm $\|g(x)\|_1$ with error $\epsilon/3$ and failure probability $\delta/3$ by using m and the norm estimation algorithm (Lemma 8.3.11).
3. Sample $i_1, \dots, i_K \sim \mathcal{D}_{g(x)}$ with failure probability $\delta/3$ by using the K -fold importance sampling algorithm (Theorem 7.5.1).
4. For each $k \in [K]$, compute $g(x)_{i_k}$ and output $\hat{g}^k = \tilde{\gamma} \operatorname{sgn}(g(x)_{i_k}) \cdot e_{i_k}$.

Algorithm 8.4: Gradient sampling, **GSample**(x, K, ϵ, δ).

Proposition 8.3.12 (GRADIENT SAMPLING). *Consider a vector $x \in [0, 1]^n$ stored in the data structure $\mathcal{D}_1(x)$. Fix an integer $K \leq n$ and two reals $0 < \epsilon, \delta < 1$. Then, the algorithm **GSample**(x, K, ϵ, δ) (Algorithm 8.4) outputs K vectors $\hat{g}^1, \dots, \hat{g}^K \in \mathbb{R}^n$ such that with probability at least $1 - \delta$, for all k , (1) \hat{g}^k is 1-sparse, (2) $\|\mathbb{E}[\hat{g}^k | \hat{g}^1, \dots, \hat{g}^{k-1}, x] - g(x)\|_1 \leq \epsilon$, (3) $\|\hat{g}^k\|_2 \leq 4$. The time complexity of the algorithm is $O((\sqrt{nK} + \sqrt{n}/\epsilon) \log(1/\delta))$.*

Proof. Let \mathcal{E} denote the event that the first three steps of Algorithm 8.4 are successful. We have $\Pr[\mathcal{E}] \geq 1 - \delta$ by a union bound. For each $k \in [K]$, the expected value of the output is $\mathbb{E}[\hat{g}^k | \hat{g}^1, \dots, \hat{g}^{k-1}, x, \mathcal{E}] = \mathbb{E}[\hat{g}^k | x, \mathcal{E}] = \sum_{i \in [n]} \frac{|g(x)_i|}{\|g(x)\|_1} \cdot \tilde{\gamma} \operatorname{sgn}(g(x)_i) e_i = \frac{\tilde{\gamma}}{\|g(x)\|_1} g(x)$. Thus, $\|\mathbb{E}[\hat{g}^k | \hat{g}^1, \dots, \hat{g}^{k-1}, x, \mathcal{E}] - g(x)\|_1 = |\tilde{\gamma} - \|g(x)\|_1| \leq (\epsilon/3) \|g(x)\|_1 \leq \epsilon$ by part (3) of Proposition 8.3.3. Moreover, \hat{g}^k is 1-sparse by definition of step 4, and $\|\hat{g}^k\|_2 = \tilde{\gamma} \leq (1 + \epsilon/3) \|g(x)\|_1 \leq 4$. Step 1 takes time $O(\sqrt{n} \log(1/\delta))$, step 2 takes time $O((\sqrt{n}/\epsilon) \log(1/\delta))$, step 3 takes time $O(\sqrt{nK} \log(1/\delta))$, and step 4 takes time $O(K)$. \square

Our second result is a more subtle gradient difference sampling algorithm **GDSample** that estimates the difference $g(y) - g(x)$ between the Lovász subgradients at two points x, y when $x \leq y$ or $x \geq y$. It uses the data structure described in Section 8.3.3. The time complexity is only $\tilde{O}(\sqrt{c}/\epsilon)$ when the difference $y - x$ is c -sparse.

Proposition 8.3.13 (GRADIENT DIFFERENCE SAMPLING). *Let c and c' be two integers such that $c' \leq 9c$. Consider two vectors $x, y \in [0, 1]^n$ and a c' -cover \mathcal{I} of (x, y) stored in the data structure $\mathcal{D}_2(x, y, \mathcal{I})$ such that $x - y$ is c -sparse, and $x \leq y$ or $x \geq y$. Fix two reals $0 < \epsilon, \delta < 1$. Then, the algorithm **GDSample**(x, y, ϵ, δ) (Algorithm 8.5) outputs a vector \tilde{d} such that with probability at least $1 - \delta$, (1) \tilde{d} is 1-sparse, (2) $\|\mathbb{E}[\tilde{d} | x, y] - (g(y) - g(x))\|_1 \leq \epsilon$, (3) $\|\tilde{d}\|_2 \leq 7$. The time complexity of the algorithm is $\tilde{O}(\sqrt{c}/\epsilon \cdot \log(1/\delta))$.*

Proof. Let \mathcal{E} denote the event that the first three steps of Algorithm 8.5 are successful. We have $\Pr[\mathcal{E}] \geq 1 - \delta$ by a union bound. The output value satisfies $\mathbb{E}[\tilde{d} | x, y, \mathcal{E}] = \sum_{j \in [c']} \frac{w_j}{\|w\|_1} \sum_{i \in \mathcal{I}_j} \frac{|g(y)_i - g(x)_i|}{w_j} \cdot \tilde{\gamma} \operatorname{sgn}(g(y)_i - g(x)_i) e_i = \frac{\tilde{\gamma}}{\|g(y) - g(x)\|_1} (g(y) - g(x))$ since $\|w\|_1 = \|g(y) - g(x)\|_1$. Thus, $\|\mathbb{E}[\tilde{d} | x, y, \mathcal{E}] - (g(y) - g(x))\|_1 \leq (\epsilon/6) \|g(y) - g(x)\|_1 \leq \epsilon$ by part (3) of Proposition 8.3.3. Moreover, $\|\tilde{d}\|_2 \leq (1 + \epsilon/6) \|g(y) - g(x)\|_2 \leq 7$. Finally,

Define the c' -dimensional vector $w = (\|g(y)_{I_j} - g(x)_{I_j}\|_1)_{j \in [c']} \in \mathbb{R}^{c'}$ to which we have a quantum oracle access by the subnorm operation of Proposition 8.3.10.

1. Compute the maximum value $m = \|w\|_\infty$ with failure probability $\delta/2$ by using the maximum finding algorithm (Proposition 7.3.1).
2. Compute an estimate $\tilde{\gamma}$ of the norm $\|w\|_1$ with error $\epsilon/6$ and failure probability $\delta/2$ by using m and the norm estimation algorithm (Lemma 8.3.11).
3. Sample $j \sim \mathcal{D}_w$ by using m and the importance sampling algorithm (Proposition 7.3.2).
4. Sample $i \sim \mathcal{D}_{g(y)_{I_j} - g(x)_{I_j}}$ by using the subsampling operation of Proposition 8.3.10.
5. Compute $g(y)_i - g(x)_i$ and output $\tilde{d} = \tilde{\gamma} \operatorname{sgn}(g(y)_i - g(x)_i) \cdot e_i$.

Algorithm 8.5: Gradient difference sampling, $\text{GDSample}(x, y, \epsilon, \delta)$.

since the subnorm operation runs in time $O(\log n)$ (Proposition 8.3.10), step 1 takes time $\tilde{O}(\sqrt{c'} \log(1/\delta))$, step 2 takes time $O((\sqrt{c'}/\epsilon) \log(1/\delta))$, step 3 takes time $O(\sqrt{c'} \log(1/\delta))$, and steps 4 and 5 take time $\tilde{O}(1)$. \square

8.3.5 Final algorithm

We combine the two procedures described in the previous section to construct a particular sequence $(\tilde{g}^{(t)})_t$ of Lovász subgradient estimates on which we apply the noisy stochastic subgradient descent method (Algorithm 8.6). The construction depends on a “loop parameter” K that balances the cost between using GSample and GDSample . Every K iterations, the procedure GSample returns K estimates $\hat{g}^{(t,0)}, \dots, \hat{g}^{(t,K-1)}$ of the Lovász subgradient at the current point $x^{(t)}$. Each value $\hat{g}^{(t,\tau)}$ is combined at time $t + \tau$, where $0 \leq \tau \leq K - 1$, with an estimate $\tilde{d}^{(t+\tau)}$ of the subgradient difference $g(x^{(t+\tau)}) - g(x^{(t)})$ computed by GDSample . The sum $\tilde{g}^{(t+\tau)} = \hat{g}^{(t,\tau)} + \tilde{d}^{(t+\tau)}$ is our estimate of $g(x^{(t+\tau)})$. The sparsity of $x^{(t+\tau)} - x^{(t)}$ increases linearly in τ , which requires to reuse GSample every K iterations to restore it to a small value. We show in the next proposition that $(\tilde{g}^{(t)})_t$ is indeed a sequence of noisy subgradients.

Proposition 8.3.14. *The sequences $(\tilde{g}^{(t)})_t$ and $(x^{(t)})_t$ defined in Algorithm 8.6 satisfy with probability at least $1 - \epsilon/4$ that (1) $\|\mathbb{E}[\tilde{g}^{(t)} \mid x^{(t)}] - g(x^{(t)})\|_1 \leq \epsilon_0 + 2\epsilon_1$, (2) $\|\tilde{g}^{(t)}\|_2 \leq 18$ and (3) $x^{(t+1)} = \arg\min_{x \in [0,1]^n} \|x - (x^{(t)} - \eta \tilde{g}^{(t)})\|_2$.*

Proof. Let \mathcal{E} denote the event that all calls to GSample and GDSample are successful in Algorithm 8.6. We have $\Pr[\mathcal{E}] \geq 1 - 2T\delta \geq 1 - \epsilon/4$ by a union bound. We carry out the analysis of the algorithm by assuming that \mathcal{E} holds. For the ease of notation, we omit to write the conditioning on \mathcal{E} below.

Fix t and $\tau = (t \bmod K)$. According to steps 3.b and 3.c of the algorithm, we have

$$\begin{cases} \tilde{g}^{(t)} = \hat{g}^{(t,0)} & \text{if } \tau = 0 \\ \tilde{g}^{(t)} = \hat{g}^{(t-\tau,\tau)} + \tilde{d}_+^{(t)} + \tilde{d}_-^{(t)} & \text{otherwise.} \end{cases}$$

1. Set $K = \lceil \epsilon \sqrt{n} \rceil$, $T = \lceil \frac{72^2 n}{\epsilon^2} \rceil$, $\epsilon_0 = \frac{\epsilon}{4}$, $\epsilon_1 = \frac{\epsilon}{8}$, $\eta = \sqrt{\frac{n}{18^2 T}}$ and $\delta = \frac{\epsilon}{8T}$.
2. Set $x^{(0)} = 0^n \in [0, 1]^n$.
3. For $t = 0, \dots, T$:
 - a) Set $\tau = (t \bmod K)$.

Computation of the subgradient estimate $\tilde{g}^{(t)}$:

 - b) If $\tau = 0$: sample $\hat{g}^{(t,0)}, \dots, \hat{g}^{(t,K-1)}$ by using the **gradient sampling** algorithm $\text{GSample}(x^{(t)}, K, \epsilon_0, \delta)$. Set $\tilde{g}^{(t)} = \hat{g}^{(t,0)}$.
 - c) If $\tau \neq 0$: sample $\tilde{d}_+^{(t)}$ by using the **gradient different sampling** algorithm $\text{GDSample}(x^{(t-\tau)}, x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, \epsilon_1, \delta)$ and $\tilde{d}_-^{(t)}$ by using the **gradient different sampling** algorithm $\text{GDSample}(x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, x^{(t)}, \epsilon_1, \delta)$. Set $\tilde{g}^{(t)} = \hat{g}^{(t-\tau, \tau)} + \tilde{d}^{(t)}$ where $\tilde{d}^{(t)} = \tilde{d}_+^{(t)} + \tilde{d}_-^{(t)}$.

Update of the position to $x^{(t+1)}$:

 - d) Compute $x^{(t+1)} = x^{(t)} + u^{(t)}$ where for each $i \in [n]$,

$$u_i^{(t)} = \begin{cases} -x_i^{(t)} & \text{if } \eta \tilde{g}_i^{(t)} > x_i^{(t)} \\ 1 - x_i^{(t)} & \text{if } \eta \tilde{g}_i^{(t)} < -(1 - x_i^{(t)}) \\ -\eta \tilde{g}_i^{(t)} & \text{otherwise.} \end{cases}$$

Update of the position difference to $z^{(t)} = x^{(t+1)} - x^{(t-\tau)}$:

 - e) If $\tau = 0$, set $z^{(t)} = u^{(t)}$.
 - f) If $\tau \neq 0$, set $z^{(t)} = z^{(t-1)} + u^{(t)}$.
4. Output $\bar{x} = \frac{1}{T} \sum_{t=0}^{T-1} x^{(t)}$.

 Algorithm 8.6: Subgradient descent algorithm for the Lovász extension f .

We first study the expectation of the term $\hat{g}^{(t-\tau, \tau)}$, which is generated by the **GSample** procedure. By the law of total expectation, we have that

$$\begin{aligned} \mathbb{E}[\hat{g}^{(t-\tau, \tau)} \mid x^{(t)}] &= \mathbb{E}\left[\mathbb{E}[\hat{g}^{(t-\tau, \tau)} \mid (\hat{g}^{(t-\tau, k)})_{k < \tau}, x^{(t-\tau)}, x^{(t)}] \mid x^{(t)}\right] \\ &= \mathbb{E}\left[\mathbb{E}[\hat{g}^{(t-\tau, \tau)} \mid (\hat{g}^{(t-\tau, k)})_{k < \tau}, x^{(t-\tau)}] \mid x^{(t)}\right] \end{aligned}$$

since $x^{(t)}$ does not convey any information about the output of **GSample**($x^{(t-\tau)}, K$) when $(\hat{g}^{(t-\tau, k)})_{k < \tau}$ and $x^{(t-\tau)}$ are known. Consequently, by using the triangle inequality and Proposition 8.3.12,

$$\begin{aligned} \|\mathbb{E}[\hat{g}^{(t-\tau, \tau)} - g(x^{(t-\tau)}) \mid x^{(t)}]\|_1 &\leq \mathbb{E}\left[\|\mathbb{E}[\hat{g}^{(t-\tau, \tau)} \mid (\hat{g}^{(t-\tau, k)})_{k < \tau}, x^{(t-\tau)}] - g(x^{(t-\tau)})\|_1 \mid x^{(t)}\right] \\ &\leq \epsilon_0. \end{aligned}$$

We now study the expectation of the term $\tilde{d}^{(t)} = \tilde{d}_+^{(t)} + \tilde{d}_-^{(t)}$ computed with the **GDSample**

procedure when $\tau \neq 0$. We have that,

$$\begin{aligned}\mathbb{E}[\tilde{d}^{(t)} \mid x^{(t)}] &= \mathbb{E}\left[\mathbb{E}[\tilde{d}_+^{(t)} \mid x^{(t-\tau)}, z_{\geq 0}^{(t-1)}, x^{(t)}] + \mathbb{E}[\tilde{d}_-^{(t)} \mid x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, z_{\leq 0}^{(t-1)}, x^{(t)}] \mid x^{(t)}\right] \\ &= \mathbb{E}\left[\mathbb{E}[\tilde{d}_+^{(t)} \mid x^{(t-\tau)}, z_{\geq 0}^{(t-1)}] + \mathbb{E}[\tilde{d}_-^{(t)} \mid x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, z_{\leq 0}^{(t-1)}] \mid x^{(t)}\right]\end{aligned}$$

where the first line is by the law of total expectation, and the second line is by independence between random variables. Moreover, by Proposition 8.3.13, $\|\mathbb{E}[\tilde{d}_+^{(t)} \mid x^{(t-\tau)}, z_{\geq 0}^{(t-1)}] - (g(x^{(t-\tau)} + z_{\geq 0}^{(t-1)}) - g(x^{(t-\tau)}))\|_1 \leq \epsilon_1$ and $\|\mathbb{E}[\tilde{d}_-^{(t)} \mid x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, z_{\leq 0}^{(t-1)}] - (g(x^{(t-\tau)} + z_{\geq 0}^{(t-1)}) - g(x^{(t-\tau)} + z_{\leq 0}^{(t-1)}))\|_1 \leq \epsilon_1$ (where we used that $x^{(t)} = x^{(t-\tau)} + z_{\geq 0}^{(t-1)} + z_{\leq 0}^{(t-1)}$). Thus, by the triangle inequality,

$$\|\mathbb{E}[\tilde{d}^{(t)} - (g(x^{(t)}) - g(x^{(t-\tau)})) \mid x^{(t)}]\|_1 \leq 2\epsilon_1.$$

This concludes the proof of the first part of the theorem since $\|\mathbb{E}[\tilde{g}^{(t)} \mid x^{(t)}] - g(x^{(t)})\|_1 = \|\mathbb{E}[\tilde{g}^{(t,0)} - g(x^{(t)}) \mid x^{(t)}]\|_1 \leq \epsilon_0$ when $\tau = 0$, and $\|\mathbb{E}[\tilde{g}^{(t)} \mid x^{(t)}] - g(x^{(t)})\|_1 \leq \|\mathbb{E}[\tilde{g}^{(t-\tau,\tau)} - g(x^{(t-\tau)}) \mid x^{(t)}]\|_1 + \|\mathbb{E}[\tilde{d}^{(t)} - (g(x^{(t)}) - g(x^{(t-\tau)})) \mid x^{(t)}]\|_1 \leq \epsilon_0 + 2\epsilon_1$ when $\tau \neq 0$. The second part of the proposition is a direct application of the triangle inequality together with $\|\tilde{g}^{(t-\tau,\tau)}\|_2 \leq 4$ and $\|\tilde{d}_+^{(t)}\|_2, \|\tilde{d}_-^{(t)}\|_2 \leq 7$ (Propositions 8.3.12 and 8.3.13). The last part of the proposition is obtained by an easy direct calculation showing that $\operatorname{argmin}_{x \in [0,1]^n} \|x - (x^{(t)} - \eta \tilde{g}^{(t)})\|_2 = x^{(t)} + u^{(t)}$. \square

The above result shows that Algorithm 8.6 is a (noisy) subgradient descent for the Lovász extension. Consequently, the result of Proposition 8.3.5 can be applied to the output \bar{x} of the algorithm. Since we aim for a subquadratic running time in n , we must update the vectors $x^{(t)}$, $\tilde{g}^{(t)}$, $u^{(t)}$ and $z^{(t)}$ in time less than their dimensions. This is done by maintaining two instances of the data structure D_2 , one for the pair $(x^{(t-\tau)}, x^{(t-\tau)} + z_{\geq 0}^{(t-1)})$ (needed for $\tilde{d}_+^{(t)}$), and the other one for the pair $(x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, x^{(t)})$ (needed for $\tilde{d}_-^{(t)}$). Since the outputs of `GSample` and `GDSample` are 1-sparse, most of the coordinates do not change between two consecutive iterations. Thus, the vectors $x^{(t)}$ and $\tilde{g}^{(t)}$ can be stored in a compact manner and the data structures can be updated at a negligible cost.

Fact 8.3.15. *At iteration t of the algorithm: $\tilde{g}^{(t)}$ and $u^{(t)}$ are 3-sparse, $z_{\geq 0}^{(t)}$ and $z_{\leq 0}^{(t)}$ are $3(\tau + 1)$ -sparse, if $\tau \neq 0$ then $z_{\geq 0}^{(t-1)}$ and $z_{\geq 0}^{(t)}$ (resp. $z_{\leq 0}^{(t-1)}$ and $z_{\leq 0}^{(t)}$) can differ only at positions where $u^{(t)}$ is non-zero.*

Proposition 8.3.16. *One can maintain throughout Algorithm 8.6 two data structures $D_2(x^{(t-\tau)}, x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, \mathcal{I})$ and $D_2(x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, x^{(t)}, \mathcal{I}')$, where \mathcal{I} and \mathcal{I}' are two covers of size at most 9τ and 27τ respectively. The update time at each iteration is $O(\log n)$.*

Proof. This is a direct consequence of Fact 8.3.15 and Proposition 8.3.10. The update from $z_{\geq 0}^{(t-1)}$ to $z_{\geq 0}^{(t)}$ (resp. $z_{\leq 0}^{(t-1)}$ to $z_{\leq 0}^{(t)}$) is 3-sparse, thus each iteration increases the size of the cover associated with $(x^{(t-\tau)}, x^{(t-\tau)} + z_{\geq 0}^{(t-1)})$ by 9, and the size of the cover associated with $(x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, x^{(t)}) = (x^{(t-\tau)} + z_{\geq 0}^{(t-1)}, x^{(t-\tau)} + z_{\geq 0}^{(t-1)} + z_{\leq 0}^{(t-1)})$ by 27. When $\tau = 0$, the sizes are reset to at most 9 and 27 respectively. \square

We execute Algorithm 8.6 with the two data structures given in Proposition 8.3.16 to obtain our final algorithm. The properties of this algorithm are summarized in the next theorem. For simplicity in the proof, we assume that the error parameter ϵ is larger than $n^{-1/6}$. This assumption is removed in [HRRS19].

Theorem 8.3.17 (SUBMODULAR MINIMIZATION). *There exists a quantum algorithm such that, given a submodular function $F : 2^V \rightarrow [-1, 1]$ and a real $\epsilon \in (\frac{1}{n^{1/6}}, 1)$, it computes a set \bar{S} such that $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ in time $\tilde{O}(n^{5/4}/\epsilon^{5/2})$.*

Proof. Let \mathcal{E} denote the event that Algorithm 8.6 is successful. We have $\Pr[\mathcal{E}] \geq 1 - \epsilon/4$ by Proposition 8.3.14. If \mathcal{E} holds then $(\tilde{g}^{(t)})_t$ is a sequence of ϵ -noisy subgradient estimate for the Lovász extension f , where $\epsilon = \epsilon_0 + 2\epsilon_1$ and $(x^{(t)})_t$ obeys the subgradient descent update rule $x^{(t+1)} = \operatorname{argmin}_{x \in [0,1]^n} \|x - (x^{(t)} - \eta \tilde{g}^{(t)})\|_2$. Moreover, $\|x - x^*\|_2 \leq \sqrt{n}$, $\|x - x^*\|_\infty \leq 1$ for all $x \in [0,1]^n$, and $\|\tilde{g}^{(t)}\|_2 \leq 18$. Consequently, by property of the noisy stochastic subgradient descent method (Proposition 8.3.5), we have that

$$\mathbb{E}[f(\bar{x}) \mid \mathcal{E}] \leq f(x^*) + 18\sqrt{n/T} + \epsilon_0 + 2\epsilon_1 \leq f(x^*) + 3\epsilon/4.$$

Thus, $\mathbb{E}[f(\bar{x})] = \Pr[\mathcal{E}] \cdot \mathbb{E}[f(\bar{x}) \mid \mathcal{E}] + (1 - \Pr[\mathcal{E}]) \cdot \mathbb{E}[f(\bar{x}) \mid \bar{\mathcal{E}}] \leq 1 \cdot (f(x^*) + 3\epsilon/4) + \epsilon/4 \cdot 1 \leq f(x^*) + \epsilon$. We can convert \bar{x} into a set $\bar{S} \subseteq V$ such that $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ with $O(n)$ queries and $O(n \log n)$ other operations by using Proposition 8.3.3.

We now analyze the run time of Algorithm 8.6 when the data structures described in Proposition 8.3.16 are maintained throughout the algorithm. The total run time of steps 3.b and 3.c is $\tilde{O}\left(\frac{T}{K}(\sqrt{nK} + \frac{\sqrt{n}}{\epsilon} + \sum_{\tau=1}^K \frac{\sqrt{\tau}}{\epsilon}) \cdot \log(1/\delta)\right) = \tilde{O}(n^{5/4}/\epsilon^{5/2})$ by Proposition 8.3.12 and Proposition 8.3.13. The vectors manipulated by the algorithm are stored in the two data structures described in Proposition 8.3.16 and in a sparse representation whose size is proportional to the number of non-zero entries. The cost of updating this representation is $O(\log n)$ at each iteration. \square

8.4 Discussion

We described other results related to the Hedge algorithm in [RHR+21], such as a quantum analog of the *Sparsitron* algorithm for learning generalized linear models or Ising models.

We do not know of any quantum algorithm that achieves a quantum speedup over the best classical algorithms for approximate [ALS20] or exact [LSW15] submodular function minimization. We do not know either of any nontrivial lower bound for these problems in the quantum query model. Classically, the best known lower bound is $\Omega(n/\log n)$ for exact minimization [Har08], which is proved by a reduction from the Connectivity problem with *cut queries* (the cut function is a submodular function). However, Lee, Santha and Zhang [LSZ21] recently showed that deciding if a graph is connected requires only $O(\log^6 n)$ quantum cut queries. They left as an open problem whether the more general Min-cut problem is hard to solve with a quantum cut oracle. A related question is to understand the power of quantum oracles that return global information (e.g. the size of a cut), as opposed to those that only reveal local information about the input (such as the general graph model used in Chapter 6). We refer the reader to [LSZ21; MS20; CHL21] for some recent results in this direction.

Part IV

Quantum Algorithms with Limited Memory

Frequency Moments and Linear Sketches in the Data Stream Model

This chapter is based on the following paper:

[HM19] Y. Hamoudi and F. Magniez. “Quantum Chebyshev’s Inequality and Applications”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 69:1–69:16.

9.1 Introduction

The *data stream* model addresses the fundamental problem of processing a large set of data that is not available for direct access, such as network traffic or a remote database. In this model, the input to a *streaming algorithm* is represented as a long and uncontrolled stream of information whose size exceeds the storage capacity of the algorithm. The elements of the stream must be processed as they arrive, which requires on-the-fly identification of the information to be kept in memory. The performances of a streaming algorithm are often measured with respect to three parameters: the *memory size* S of the algorithm, the number P of *passes* that can be made over the stream, and the *update time* T spent on each element. The interplay between these three quantities can change drastically if the streaming algorithm is equipped with a *quantum memory*. Indeed, the ability to store information on quantum bits can sometimes decrease the memory size requirement by a polynomial [Mon16] or an exponential [Gal09; GKK+09; Mon11] factor in the input length. Such algorithms are of great interest to understand the power of small quantum computers. Nevertheless, very few quantum algorithmic techniques are known in the data stream model due to the sequential access to the input. In this chapter, we describe a new tool for designing quantum streaming algorithms, and we apply it to the problem of estimating the *frequency moments* of a stream.

The most studied problems in the streaming literature take place in the *turnstile model*, where a vector x that is initially set to 0^n is incremented by a stream of updates $x_i \leftarrow x_i + \lambda$ described by the pairs $(i, \lambda) \in [n] \times \mathbb{Z}$. The goal is to compute some statistics of the final vector x by using much less memory than it would take to store x entirely. Remarkably, under certain conditions, one can show [LNW14; AHLW16] that any algorithm in this setting can be implemented efficiently by maintaining in memory a *linear sketch* $M \cdot x$ of the current vector x , for some input-independent matrix M . We raise the question to what extent such linear sketch algorithms can be used as *subroutines* of quantum streaming algorithms. It is well known [Ben73; Ben89] that any classical algorithm can be made reversible, and therefore implemented by a unitary map U . Nevertheless, as noticed by Montanaro [Mon16], the standard reversibility techniques may not be space-efficient in the

data stream model. We exploit the properties of the linear sketch algorithms to circumvent this problem, and we describe a general quantum streaming algorithm for simulating a unitary U that encodes the linear sketch, and its inverse U^{-1} , in a space-efficient way.

We apply our simulation result to the problem of estimating the *frequency moment* $F_k(x) = \sum_{i \in [n]} |x_i|^k$ of order $k \geq 3$ with relative error $\epsilon \in (0, 1)$ in the turnstile model. This question and its variants have been studied since the early ages of the data stream model [Mor78; FM85; Fla85; AMS99], and it led to the development of some of the most fundamental techniques in the streaming literature [AMS99; BJKS04; IW05; MW10]. The optimal memory size S needed to estimate $F_k(x)$ with P passes in the classical setting is $S = \tilde{\Theta}(n^{1-2/k}/P)$ [MW10; WZ12]. We describe a new quantum streaming algorithm that uses a smaller memory of size $S = \tilde{O}(n^{1-2/k}/P^2)$. Our approach is based on a particular type of streaming algorithms, called the L_p *samplers*, that received a lot of interest in recent years [CJ19]. An L_p sampler returns an index $i \in [n]$ with a probability that is proportional to the value of $|x_i|^p$. We use our simulation result on a linear sketch L_2 sampler [MW10; AKO10] to construct a q-random variable that estimates the value of $F_k(x)$. We then adapt the quantum sub-Gaussian estimator of Chapter 4 to the multi-pass data stream model in order to estimate the expectation of that q-random variable.

9.1.1 Related work

The problem of approximating the frequency moment $F_k(x)$ of order $k \geq 3$ has been studied first in the seminal work of Alon, Matias, and Szegedy [AMS99]. The authors described a classical single-pass streaming algorithm using $O(n^{1-1/k} \log n)$ memory bits (where we omit the dependence on ϵ) in the positive-update model. A long series of improvements in the more general turnstile model culminated into the optimal memory size of $\Theta(n^{1-2/k} \log n)$ [LW13; Gan15] for classical single-pass algorithms, and the nearly optimal pass-memory tradeoff of $PS = \tilde{\Theta}(n^{1-2/k})$ [MW10; AKO10; WZ12] for classical P -pass algorithms with memory size S .

The frequency moments have received little attention in the quantum streaming model. Montanaro [Mon16] described an algorithm that uses a quantum memory of size $S = O(\log n + \log(n^{1-1/k}/\epsilon))$ and that makes $P = \tilde{O}(n^{1-1/k}/\epsilon)$ passes over the stream. This is better than the best classical streaming algorithm with the same number of passes when ϵ is sufficiently small. He also proved a pass-memory lower bound of $PS \geq \Omega(1/\epsilon + \log n)$ when $\epsilon \geq 1/\sqrt{n}$. Jain, Radhakrishnan and Sen [JRS03] proved two lower bounds for the \mathcal{L}_∞ and the multi-party Set Disjointness problems in the bounded-round quantum communication model. Their results imply that $PS \geq \Omega\left(\frac{n^{1-2/k}}{P^3 n^{(P-1)/k}}\right)$ and $PS \geq \Omega\left(\frac{n^{1-2/k}}{P^2 n^{1/k}}\right)$ (where we omit the dependence on ϵ) by a standard reduction [BJKS04] to the Frequency Moments problem. In particular, for $P = 1$, the lower bound $S \geq \Omega(n^{1-2/k})$ implies that no quantum speedup is possible in the single-pass regime. Finally, the related problem of approximating the frequency moments (or the Rényi entropy) in the quantum query model has been studied in [Mon16; LW19].

There are a few other problems that have been considered in the quantum streaming model. Le Gall [Gal09] proved a pass-memory tradeoff of $P^2 S = \tilde{\Theta}(n)$ for a variant of the DISJOINTNESS problem, whereas the best classical streaming algorithm satisfies $PS = \Theta(n)$. Similarly, for the DYCK(2) problem, there is a pass-memory lower bound of $P^3 S = \Omega(\sqrt{n})$ [NT17] in the quantum model, and a tradeoff of $PS = \tilde{\Theta}(\sqrt{n})$ [MMN14; CCKM13; JN14] in the classical one. Finally, there exist exponential separations [Gal09; GKK+09; Mon11] in memory size between the classical and the quantum single-pass models for some artificial problems.

9.1.2 Contributions and organization

We first present the *turnstile* variant of the data stream model in Section 9.2, and we define what a randomized or quantum streaming algorithm is. Next, we introduce the notion of *reversible streaming algorithm* in Section 9.3.1 and we prove that the computation performed by such an algorithm can be represented as a unitary transformation U such that both U and U^{-1} can be simulated efficiently by a quantum streaming algorithm (Proposition 9.3.3). We show in Section 9.3.2 that any linear sketch algorithm can be turned into a reversible streaming algorithm with a limited overhead on memory.

We apply the above simulation results to the problem of estimating the frequency moment $F_k(x)$ in Section 9.4. We first describe a classical linear sketch algorithm for estimating $F_k(x)$ (Proposition 9.4.2) based on an L_2 sampler (Proposition 9.4.1). We then turn this algorithm into a particular quantum streaming algorithm that allows us to estimate $F_k(x)$ by using the quantum sub-Gaussian estimator of Chapter 4. Our final result is summarized in the next theorem.

Theorem 9.4.3 (Restated). *For any integer $P \geq 1$, there exists a P -pass streaming algorithm with the following properties. Given an input stream in the turnstile model with non-zero final vector $x \in \mathbb{Z}^n$, an integer $k \geq 3$ and a real $\epsilon \in (\frac{10}{n}, 1)$, the algorithm uses a quantum memory of size*

$$\tilde{O}\left(\frac{n^{1-2/k}}{(\epsilon P)^2}\right)$$

and it outputs an estimate \tilde{F}_k such that $|\tilde{F}_k - F_k(x)| \leq \epsilon F_k(x)$ with probability at least $3/4$.

9.1.3 Proof overview

Our starting point is the following *classical* streaming algorithm for estimating the frequency moment $F_k(x) = \sum_{i \in [n]} |x_i|^k$, which is optimal up to a polylogarithmic factor. Consider the unbiased estimator of $F_k(x)$ that takes value $F_2(x)|x_i|^{k-2}$ with probability $\frac{|x_i|^2}{F_2(x)}$ for each $i \in [n]$. The so-called L_2 sampler algorithms [MW10; AKO10; CJ19] can output one (approximate) sample from this estimator by doing one pass over the input stream and by using $\text{polylog } n$ bits of memory (Proposition 9.4.1). The empirical mean \tilde{F}_k of $O(n^{1-2/k}/\epsilon^2)$ such samples satisfies the objective bound $|\tilde{F}_k - F_k(x)| \leq \epsilon F_k(x)$ with high probability by Chebyshev's inequality (Proposition 9.4.2). The computation of \tilde{F}_k can be distributed over P passes, for any value of $P \geq 1$, which leads to a streaming algorithm with memory size $\tilde{O}\left(\frac{n^{1-2/k}}{P}\right)$ (where we omit the dependence on ϵ). In comparison, the seminal AMS algorithm [AMS99] on which is based Montanaro's work [Mon16] uses an L_1 sampler to obtain $F_1(x)|x_i|^{k-1}$ with probability $\frac{|x_i|}{F_1(x)}$. The latter estimator is easier to implement but it has a larger variance.

The quantum sub-Gaussian estimator developed in Chapter 4 (Theorem 4.4.2) can estimate the expectation of a random variable X quadratically faster than it is possible with the empirical mean estimator, under certain conditions. The main requirement is to have access to a unitary transformation U , and its inverse U^{-1} , such that $U|0\rangle$ encodes the distribution of X in superposition. In order to apply the quantum sub-Gaussian estimator to the random variable that takes value $F_2(x)|x_i|^{k-2}$ with probability $\frac{|x_i|^2}{F_2(x)}$, we transform a classical L_2 sampler [AKO10] into a unitary algorithm U such that *one use* of U or U^{-1} can be simulated by a quantum streaming algorithm that makes *one pass* over the input stream. To this end, it is well known that any classical circuit can be made reversible, and

therefore represented by a unitary map, with a certain overhead on the time and space complexities [Ben73; Ben89]. Nevertheless, the sequential nature of the data stream model raises some additional challenges here. First, the use of garbage bits to store the history of the computation as in [Ben73] can cause the size of the memory to grow linearly with the length of the stream. The recursive simulation technique described in [Ben89] is space efficient, but it would require breaking the stream into segments that must be repeated many times. Secondly, the *reverse* computation of a reversible circuit is usually done by running the circuit backward. In the case of a streaming algorithm, it would require processing the input stream in the *reverse* direction, which is not permitted by our model.

We address these issues by using the fact that the L_2 sampler described in [AKO10] is a *linear sketch* algorithm. A linear sketch algorithm (Section 9.2) is a streaming algorithm with the additional property that the content of the memory depends *linearly* on the input stream. In particular, the final memory of a linear sketch algorithm is invariant under any permutation of the order of arrival of the stream. We prove that any such algorithm can be made reversible efficiently (Proposition 9.3.5), which allows us to simulate the related unitary transformation with a quantum streaming algorithm (Proposition 9.3.3). We finally use the quantum sub-Gaussian estimator on this unitary to estimate $F_k(x)$ (Theorem 9.4.3). We cannot use the full power of the sub-Gaussian estimator when the number P of passes is small (since one pass over the stream allows us to perform only one quantum experiment). In this case, we first decrease the variance of the original random variable by using the empirical mean estimator (Proposition 9.4.2).

9.2 Data stream model

We refer the reader to [Mut05] for a general introduction to the classical data stream model. In this chapter, we consider the data stream model with general updates (also called the *turnstile model*), where the objective is to compute some function $f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ of a vector $x \in \mathbb{Z}^n$ that is described by a stream $u^{(1)}, u^{(2)}, \dots, u^{(t)}$ of t updates. An update is a pair $(i, \lambda) \in [n] \times \mathbb{Z}$ that represents the addition of a value λ to the i -th coordinate of x . More formally, we denote by $x^{(0)} \in \mathbb{Z}^n$ the all-0 vector, and we define the *intermediate vector* $x^{(j)}$ for $j \in [t]$ as

$$x^{(j)} = x^{(j-1)} + \lambda_j e_{i_j} \quad (9.1)$$

where $u^{(j)} = (i_j, \lambda_j)$ is the j -th update of the stream and $e_{i_j} \in \mathbb{Z}^n$ is the indicator vector with a 1 at position i_j . The input to the function f is the *final vector* $x = x^{(t)}$. We assume that there exists a universal constant p such that the length t of the stream is at most n^p , and the intermediate vectors satisfy $x^{(j)} \in \{-n^p, -n^p + 1, \dots, n^p\}^n$ at all times of the stream. The space bounds are measured in terms of the number of (qu)bits of memory.

Randomized streaming algorithm. We describe a particular type of randomized streaming algorithm, where the random bits are placed in the memory of the algorithm at the beginning of the computation. Given three integers S , S_{seed} and S_{comp} such that $S = S_{\text{seed}} + S_{\text{comp}}$, a *randomized streaming algorithm* with *memory size* S is defined as a family $\mathcal{A} = \{\mathcal{A}_r^{\text{in}}\}_r \cup \{\mathcal{A}_{r,u}^{\text{upd}}\}_{r,u} \cup \{\mathcal{A}_r^{\text{out}}\}_r$ of classical deterministic circuits operating on S_{comp} bits, where the index r ranges over the set $\{0, 1\}^{S_{\text{seed}}}$ of random seeds, and the index u ranges over the set $[n] \times \mathbb{Z}$ of updates. The memory of \mathcal{A} is partitioned into two parts: a seed register of size S_{seed} and a computation register of size S_{comp} . Given an integer $P \geq 1$, the P -pass randomized streaming algorithm \mathcal{A} processes a stream $u^{(1)}, \dots, u^{(t)}$ as follows.

1. (*Preprocessing*) Before the stream arrives, the seed register is filled with a random string r , the circuit $\mathcal{A}_r^{\text{in}}$ is applied to the computation register filled with zeros.
2. (*Update*) The algorithm receives the t updates of the stream in a chronological order *repeated P times* (corresponding to P passes). For each update u , the circuit $\mathcal{A}_{r,u}^{\text{upd}}$ is applied to the computation register.
3. (*Postprocessing*) At the end of the stream, the circuit $\mathcal{A}_r^{\text{out}}$ is applied to the computation register. The output is contained in some predetermined bits of the register.

The *update time* of \mathcal{A} is the largest gate complexity of a circuit in the family \mathcal{A} (including the circuits used for the preprocessing and postprocessing steps).

Quantum streaming algorithm. A *quantum streaming algorithm* with memory size S is similarly defined as a family $\mathcal{Q} = \{\mathcal{Q}^{\text{in}}\} \cup \{\mathcal{Q}_u^{\text{upd}}\}_u \cup \{\mathcal{Q}^{\text{out}}\}$ of quantum circuits operating on S qubits (see Figure 9.1). There is no seed register and seed index r since the algorithm can prepare its own random bits by measuring the $|+\rangle$ state. The input stream is processed in the same way as in the classical setting. The output is obtained by measuring some predetermined qubits of the final memory in the computational basis. We say that \mathcal{Q} is a *unitary streaming algorithm* if all circuits in \mathcal{Q} are unitary circuits (i.e. they are only made of unitary quantum gates, with no intermediate measurements).

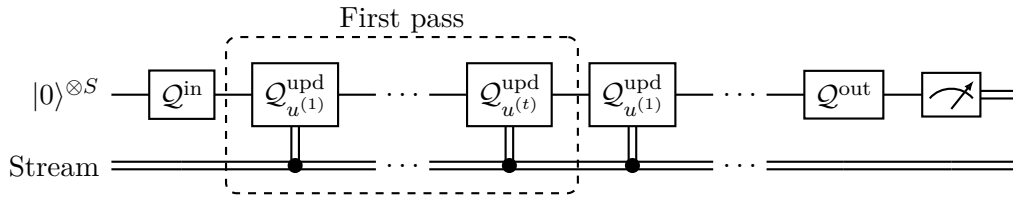


Figure 9.1: An illustration of a multi-pass quantum streaming algorithm with memory size S . The choice of which update circuits $\mathcal{Q}_u^{\text{upd}}$ to apply is controlled on the input stream (represented by the double line).

9.3 Quantum simulation of classical streaming algorithms

We study the simulation of two types of classical streaming algorithms by quantum unitary streaming algorithms. In this section, we only consider single-pass algorithms.

9.3.1 Reversible streaming algorithms

We first consider the simulation of classical streaming algorithms having reversible properties. We want both the original algorithm and its inverse to be efficiently simulated by a unitary streaming algorithm. The inverse will be useful later on to apply quantum subroutines (such as the quantum sub-Gaussian estimator of Chapter 4). We use the following definitions regarding classical deterministic circuits.

Definition 9.3.1. We consider the classical counterpart of the quantum circuit model defined in Section 3.1. Given a deterministic circuit \mathcal{C} operating on a memory of size S , we let $\mathcal{C}(m) \in \{0, 1\}^S$ denote the state of the final memory when \mathcal{C} is applied to the initial memory $m \in \{0, 1\}^S$. We say that \mathcal{C} is *reversible* if it only uses reversible elementary gates

(e.g. the Toffoli gate), and we let \mathcal{C}^{-1} denote the inverse circuit obtained by running \mathcal{C} backward. Given two circuits \mathcal{C} and \mathcal{C}' operating on the same memory space, we let $\mathcal{C} \parallel \mathcal{C}'$ denote the concatenated circuit that runs \mathcal{C} first and then \mathcal{C}' .

The computation performed by a (single-pass) classical streaming algorithm \mathcal{A} with respect to a stream $u^{(1)}, \dots, u^{(t)}$ and a random seed r corresponds to the circuit

$$\mathcal{A}_r^{\text{in}} \parallel \mathcal{A}_{r,u^{(1)}}^{\text{upd}} \parallel \dots \parallel \mathcal{A}_{r,u^{(t)}}^{\text{upd}} \parallel \mathcal{A}_r^{\text{out}}. \quad (9.2)$$

If each circuit in \mathcal{A} is reversible, then the reverse circuit of the entire computation is

$$(\mathcal{A}_r^{\text{out}})^{-1} \parallel (\mathcal{A}_{r,u^{(t)}}^{\text{upd}})^{-1} \parallel \dots \parallel (\mathcal{A}_{r,u^{(1)}}^{\text{upd}})^{-1} \parallel (\mathcal{A}_r^{\text{in}})^{-1}. \quad (9.3)$$

It is generally unclear whether such a computation can be implemented efficiently as a streaming algorithm running on the same input stream $u^{(1)}, \dots, u^{(t)}$ (it would be easy if the stream arrives in the *reverse* order). This motivates our following notion of *reversible streaming algorithm*, where the reverse computation does not require processing the stream in the reverse order. We include garbage bits in the definition since they are needed in most reversible simulation techniques. As is standard, these bits must be restored to 0 at the end of the computation.

Definition 9.3.2 (REVERSIBLE STREAMING ALGORITHM). A classical streaming algorithm \mathcal{A} is *reversible* if it satisfies the following conditions. Let S_{comp} be the size of the computation register of \mathcal{A} . There exist two integers $c, g \geq 0$ such that $S_{\text{comp}} = c + g$ and,

- (1) (*Step reversible*) Each circuit in \mathcal{A} is reversible.
- (2) (*Garbage restoring*) For any circuit \mathcal{C} in the family \mathcal{A} and any memory $m \in \{0, 1\}^c$, we have $\mathcal{C}(m, 0^g) = (m', 0^g)$ for some $m' \in \{0, 1\}^c$ that depends on \mathcal{C} and m .
- (3) (*Order independent*) For any two updates u, v , any seed r and any memory $m \in \{0, 1\}^c$, we have $(\mathcal{A}_{r,u}^{\text{upd}} \parallel \mathcal{A}_{r,v}^{\text{upd}})(m, 0^g) = (\mathcal{A}_{r,v}^{\text{upd}} \parallel \mathcal{A}_{r,u}^{\text{upd}})(m, 0^g)$.

The state of a reversible streaming algorithm is invariant under any permutation of the order of arrival of the stream (if the garbage bits are initially set to 0). Thus, the circuit of Equation (9.3) computes the same output as $(\mathcal{A}_r^{\text{out}})^{-1} \parallel (\mathcal{A}_{r,u^{(1)}}^{\text{upd}})^{-1} \parallel \dots \parallel (\mathcal{A}_{r,u^{(t)}}^{\text{upd}})^{-1} \parallel (\mathcal{A}_r^{\text{in}})^{-1}$ if \mathcal{A} is reversible. The latter circuit can be implemented by processing the input stream in the *direct* order. As a result, we show that the computation performed by a reversible streaming algorithm can be represented as a unitary transformation U such that one use of U or U^{-1} is simulated by a unitary streaming algorithm making one pass over the stream.

Proposition 9.3.3. *Given a reversible streaming algorithm \mathcal{A} with memory size S , there exist two integers $q, g \geq 0$ such that $S = q + g$ and two unitary streaming algorithms $\mathcal{Q}, \overline{\mathcal{Q}}$ with memory size S that satisfy the following conditions. For any stream $u^{(1)}, \dots, u^{(t)}$ there is a unitary operator U acting on q qubits such that,*

- (1) *If $(U|0^q\rangle)|0^g\rangle$ is measured in the computational basis, then the measurement outcome follows the same distribution as the final memory of \mathcal{A} when it is run on $u^{(1)}, \dots, u^{(t)}$.*
- (2) *If \mathcal{Q} is run on the stream $u^{(1)}, \dots, u^{(t)}$ with the initial memory being $|\psi\rangle|0^g\rangle \in \mathbb{C}^{2^S}$, then the state of the memory before applying the final measurement is $(U|\psi\rangle)|0^g\rangle$.*
- (3) *If $\overline{\mathcal{Q}}$ is run on the stream $u^{(1)}, \dots, u^{(t)}$ with the initial memory being $|\psi\rangle|0^g\rangle \in \mathbb{C}^{2^S}$, then the state of the memory before applying the final measurement is $(U^{-1}|\psi\rangle)|0^g\rangle$.*

Proof. Let \mathcal{A} be a reversible streaming algorithm with memory size S . Let S_{seed} and S_{comp} denote the size of the random seed and computation registers respectively. Let $c, g \geq 0$ be the two integers provided by Definition 9.3.2 such that $S_{\text{comp}} = c + g$. We set $q = S_{\text{seed}} + c$.

We define the unitary streaming algorithm \mathcal{Q} as follows. First, we extend by linearity the circuits in \mathcal{A} to unitary transformations controlled on a seed register $|r\rangle$. Formally, we define the quantum circuits $\hat{\mathcal{Q}}^{\text{in}}, \mathcal{Q}_u^{\text{upd}}, \mathcal{Q}^{\text{out}}$ that act on each basis state $|r\rangle|m\rangle \in \mathbb{C}^{2^{S_{\text{seed}}}} \otimes \mathbb{C}^{2^{S_{\text{comp}}}}$ as follows,

$$\hat{\mathcal{Q}}^{\text{in}}(|r\rangle|m\rangle) = |r\rangle|\mathcal{A}_r^{\text{in}}(m)\rangle, \mathcal{Q}_u^{\text{upd}}(|r\rangle|m\rangle) = |r\rangle|\mathcal{A}_{r,u}^{\text{upd}}(m)\rangle, \mathcal{Q}^{\text{out}}(|r\rangle|m\rangle) = |r\rangle|\mathcal{A}_r^{\text{out}}(m)\rangle.$$

Next, we let \mathcal{Q}^{in} denote the quantum circuit that first applies a Hadamard transform to the first S_{seed} qubits of the memory, and then that applies $\hat{\mathcal{Q}}^{\text{in}}$ to the entire memory. The obtained unitary streaming algorithm $\mathcal{Q} = \mathcal{Q}^{\text{in}} \cup \{\mathcal{Q}_u^{\text{upd}}\}_u \cup \{\mathcal{Q}^{\text{out}}\}$ is represented in Figure 9.2.

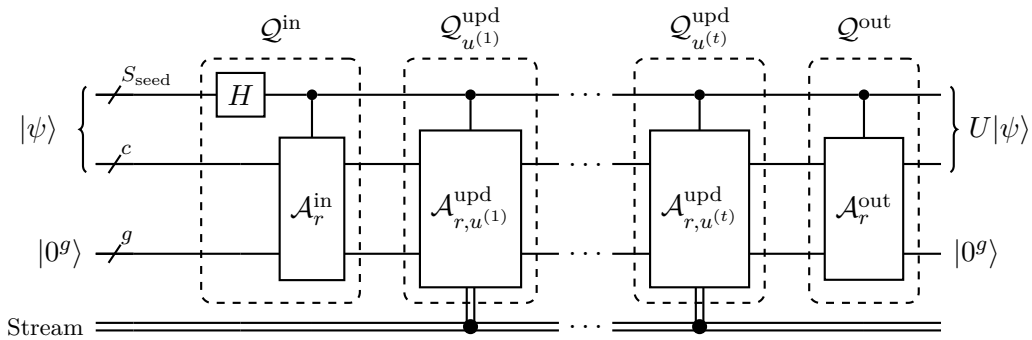


Figure 9.2: An illustration of the unitary streaming algorithm \mathcal{Q} implementing U .

Given a stream $u^{(1)}, \dots, u^{(t)}$, we let U denote the operation applied by the unitary circuit $\mathcal{Q}^{\text{in}} \parallel \mathcal{Q}_{u^{(1)}}^{\text{upd}} \parallel \dots \parallel \mathcal{Q}_{u^{(t)}}^{\text{upd}} \parallel \mathcal{Q}^{\text{out}}$ on the first q qubits of the memory when the last g qubits are initially set to 0. Part (2) of the proposition is a direct consequence of this definition, and part (1) is obtained by observing that,

$$(U|0^q\rangle)|0^g\rangle = \frac{1}{2^{S_{\text{seed}}/2}} \sum_{r \in \{0,1\}^{S_{\text{seed}}}} |r\rangle \left((\mathcal{A}_r^{\text{in}} \parallel \mathcal{A}_{r,u^{(1)}}^{\text{upd}} \parallel \dots \parallel \mathcal{A}_{r,u^{(t)}}^{\text{upd}} \parallel \mathcal{A}_r^{\text{out}}) (0^{S_{\text{comp}}}) \right).$$

The unitary streaming algorithm $\bar{\mathcal{Q}}$ is defined by taking $\bar{\mathcal{Q}}^{\text{in}} = (\mathcal{Q}^{\text{out}})^{-1}$, $\bar{\mathcal{Q}}_u^{\text{upd}} = (\mathcal{Q}_u^{\text{upd}})^{-1}$ and $\bar{\mathcal{Q}}^{\text{out}} = (\bar{\mathcal{Q}}^{\text{in}})^{-1}$. The part (3) of the proposition is deduced from the observation that $\bar{\mathcal{Q}}_u^{\text{upd}}(|r\rangle|m\rangle) = |r\rangle|(\mathcal{A}_{r,u}^{\text{upd}})^{-1}(m)\rangle$ for any basis state $|r\rangle|m\rangle$, and from the order independent property of the reversible streaming algorithms. \square

9.3.2 Linear sketch algorithms

A *linear sketch* algorithm is a particular type of randomized streaming algorithm that updates its memory according to some linear function. Most of the algorithms known in the turnstile model are linear sketches [LNW14; AHLW16]. We show that any such algorithm can be turned into a reversible streaming algorithm with a limited overhead on the space complexity. As a result, the computation performed by a linear sketch algorithm can be efficiently simulated by a unitary streaming algorithm.

We first define what a linear sketch is. Here, we interpret the S_{comp} bits contained in the computation register of the algorithm as the binary encoding of an integer-valued vector $m \in \mathbb{Z}_{2^w}^s$ with word size w , where w, s are two fixed integers such that $S_{\text{comp}} = ws$.

Definition 9.3.4 (LINEAR SKETCH). A classical streaming algorithm \mathcal{A} with memory size $S = S_{\text{seed}} + S_{\text{comp}}$ is a *linear sketch* if there exist two integers w, s such that $S_{\text{comp}} = ws$ and a family $\{M_r\}_r$ of matrices $M_r \in \mathbb{Z}_{2^w}^{s \times n}$ that satisfy the following property. Given a vector $m \in \mathbb{Z}_{2^w}^s$ loaded in the computation register, the result of applying $\mathcal{A}_{r,u}^{\text{upd}}$ to m is

$$\mathcal{A}_{r,u}^{\text{upd}}(m) = m + M_r \cdot (\lambda e_i) \bmod 2^w$$

for any seed $r \in \{0, 1\}^{S_{\text{seed}}}$ and any update $u = (i, \lambda) \in [n] \times \mathbb{Z}$.

The use of a preprocessing circuit $\mathcal{A}_r^{\text{in}}$ is unnecessary here. By linearity, the state of the computation register is $M_r \cdot x^{(j)} \bmod 2^w$ after the linear sketch algorithm has processed the first j updates of the stream (where $x^{(j)}$ is the intermediate vector defined in Equation (9.1)). We use the linearity property to transform any linear sketch algorithm into a reversible streaming algorithm.

Proposition 9.3.5. *Any linear sketch algorithm with memory size S and update time T can be converted into a reversible streaming algorithm with memory size $O(S \log(T))$ and update time $O(T^2)$ that computes the same function as the original algorithm.*

Proof. Let \mathcal{A} be a linear sketch algorithm with memory size $S = S_{\text{seed}} + S_{\text{comp}}$ and update time T . Consider the family $\{M_r\}_r$ of matrices $M_r \in \mathbb{Z}_{2^w}^{s \times n}$ provided by Definition 9.3.4 with respect to \mathcal{A} . Note that the transformation performed by each update circuit $\mathcal{A}_{r,u}^{\text{upd}}$ is bijective. Thus, we can apply the *input-erasing* reversibility technique from [Ben89, Theorem 2] to obtain a new family of reversible circuits $\mathcal{B}_{r,u}^{\text{upd}}$ that operate on a computation register of size $S_{\text{comp}} + g$ with $g = O(S \log T)$ garbage bits such that,

$$\mathcal{B}_{r,u}^{\text{upd}}(m, 0^g) = (\mathcal{A}_{r,u}^{\text{upd}}(m) \bmod 2^w, 0^g)$$

for any $m \in \mathbb{Z}_{2^w}^s$. We have $(\mathcal{B}_{r,u}^{\text{upd}} \parallel \mathcal{B}_{r,v}^{\text{upd}})(m, 0^g) = (m + M_r \cdot (\lambda e_i + \lambda' e_{i'}) \bmod 2^w, 0^g)$ for any two updates $u = (i, \lambda)$ and $v = (i', \lambda')$ by Definition 9.3.4. The right-hand side of this expression is unchanged if we permute the updates u and v . Thus, the update circuits $\mathcal{B}_{r,u}^{\text{upd}}$ satisfy the order independent property of Definition 9.3.2.

The postprocessing circuits $\mathcal{A}_r^{\text{out}}$ are not necessarily bijective. Thus, we instead use the *input-saving* reversibility technique from [Ben89, Theorem 1] to obtain a family of reversible circuits $\mathcal{B}_r^{\text{out}}$ that operate on a computation register of size $2S_{\text{comp}} + g$ with $g = O(S \log T)$ garbage bits such that,

$$\mathcal{B}_r^{\text{out}}(m', m, 0^g) = (m' \oplus \mathcal{A}_r^{\text{out}}(m) \bmod 2^w, m, 0^g)$$

for any $m, m' \in \mathbb{Z}_{2^w}^s$ (the original input m is placed in the second coordinate here). We extend the update circuits $\mathcal{B}_{r,u}^{\text{upd}}$ to act as the identity on the first S_{comp} bits of that new computation register, and we take $\mathcal{B}_{r,u}^{\text{in}}$ to be the identity since no preprocessing circuit is needed for linear sketch algorithms. The streaming algorithm $\mathcal{B} = \{\mathcal{B}_r^{\text{in}}\}_r \cup \{\mathcal{B}_{r,u}^{\text{upd}}\}_{r,u} \cup \{\mathcal{B}_r^{\text{out}}\}_r$ satisfies the conditions of Definition 9.3.2 with $q = 2S_{\text{comp}}$ and g , hence it is reversible. Moreover, the first S_{comp} bits of its computation register contain the same state as that of the original streaming algorithm \mathcal{A} at the end of the stream. \square

9.4 Estimation of the frequency moments

We describe our quantum streaming algorithm for approximating the frequency moment $F_k(x) = \sum_{i \in [n]} |x_i|^k$ of order $k \geq 3$ in the turnstile model. The algorithm combines the quantum sub-Gaussian estimator constructed in Chapter 4 with the following linear sketch L_2 sampler from Andoni, Krauthgamer and Onak [AKO10].

Proposition 9.4.1 (L_2 SAMPLER – Theorem 5.1 in [AKO10]). *There exists a single-pass linear sketch algorithm with the following properties. Given a stream of updates with non-zero final vector $x \in \mathbb{Z}^n$ and two reals $\epsilon, \delta \in (0, 1)$, the algorithm outputs with probability at least $1 - \delta$ a pair $(i, \tilde{f}_i) \in [n]$ such that $|\tilde{f}_i - |x_i|| \leq \epsilon |x_i|$ and*

$$\Pr[i = j] = (1 \pm \epsilon) \frac{|x_i|^2}{F_2(x)} \pm \frac{1}{n^2} \quad \text{for all } j \in [n],$$

and it outputs FAIL otherwise. The memory size of the algorithm is $O(\epsilon^{-2} \log^4(n) \log(1/\delta))$ and the update time is $\tilde{O}(\epsilon^{-1} n \log(1/\delta))$.

We convert the above L_2 sampler into a single-pass linear sketch algorithm for approximating $F_k(x)$ (Algorithm 9.3) by using the unbiased estimator [MW10; AKO10] that takes value $F_2(x)|x_i|^{k-2}$ with probability $\frac{|x_i|^2}{F_2(x)}$. The variance of the estimator is traded off against the memory size of the algorithm by computing several samples in parallel. For simplicity, we assume that the error parameter is $\epsilon \geq \Omega(1/n)$. The second moment $F_2(x)$ needed by the estimator can be efficiently estimated with [AMS99; KNW10] for instance.

1. Run in parallel C copies of the L_2 sampler of Proposition 9.4.1 on the same stream $u^{(1)}, \dots, u^{(t)}$ with input parameters $\frac{\epsilon}{20k}$ and $\delta = \frac{\epsilon}{2}$. For each copy $j = 1, \dots, C$:
 - a) If the algorithm returns a pair (i, \tilde{f}_i) then set $F^{(j)} = \tilde{F}_2 \cdot \tilde{f}_i^{k-2}$.
 - b) If the algorithm returns FAIL then set $F^{(j)} = 0$.
2. Output $F = \frac{1}{C} \sum_{j \in [C]} F^{(j)}$.

Algorithm 9.3: Classical frequency moment estimator.

Proposition 9.4.2. *There exists a single-pass linear sketch algorithm (Algorithm 9.3) with the following properties. Let $u^{(1)}, \dots, u^{(t)}$ be an input stream with non-zero final vector $x \in \mathbb{Z}^n$, and fix as parameters two integers $C \geq 1$, $k \geq 3$, a real $\epsilon \in (\frac{5}{n}, 1)$ and an estimate \tilde{F}_2 such that $|\tilde{F}_2 - F_2(x)| \leq \frac{\epsilon}{8} F_2(x)$. Then, the output of the algorithm is distributed according to a random variable F such that,*

- (1) $\mathbb{E}[F] \in [(1 - \epsilon)F_k(x), (1 + \epsilon)F_k(x)]$ and $\text{Var}[F] \leq \frac{2n^{1-2/k}F_k(x)^2}{C}$.
- (2) *The memory size is $O(C\epsilon^{-2} \log^4(n) \log(1/\delta))$, the update time is $\tilde{O}(C\epsilon^{-1} n \log(1/\delta))$.*

Proof. We first compute the expectation of the random variable $F^{(j)}$ obtained at step 1 for any j . If the L_2 sampler does not fail then the expectation of $F^{(j)}$ is in the interval $\mathbb{E}[F^{(j)} \mid \neg \text{FAIL}] = (1 \pm \frac{\epsilon}{8})(1 \pm \frac{\epsilon}{20k})^{k-1} \sum_i F_2(x)|x_i|^{k-2} \left(\frac{|x_i|^2}{F_2(x)} \pm \frac{1}{n^2} \right)$ by Proposition 9.4.1. Since $F_2(x)|x_i|^{k-2} \leq nF_k(x)$ and $\epsilon > 5/n$, we get that $\mathbb{E}[F^{(j)} \mid \neg \text{FAIL}] = (1 \pm \frac{\epsilon}{2})F_k(x)$.

The algorithm outputs FAIL with probability at most $\frac{\epsilon}{2}$, in which case we have $F^{(j)} = 0$. Thus, $\mathbb{E}[F^{(j)}] \leq \mathbb{E}[F^{(j)} \mid \neg \text{FAIL}]$ and $\mathbb{E}[F^{(j)}] \geq (1 - \frac{\epsilon}{2})\mathbb{E}[F^{(j)} \mid \neg \text{FAIL}] \geq (1 - \epsilon)F_k(x)$. We conclude that $|\mathbb{E}[F^{(j)}] - F_k(x)| \leq \epsilon F_k(x)$. By linearity of expectation, $\mathbb{E}[F] = \mathbb{E}[F^{(j)}]$.

The variance is upper bounded in a similar way. For any $j \in [C]$ we have, $\text{Var}[F^{(j)}] \leq \mathbb{E}[(F^{(j)})^2 \mid \neg \text{FAIL}] \leq 2F_2(x)F_{2k-2}(x) \leq 2n^{1-2/k}F_k(x)^2$, where the last step is by Hölder's inequality. Finally, we have $\text{Var}[F] = \frac{1}{C} \text{Var}[F^{(j)}] = \frac{2n^{1-2/k}F_k(x)^2}{C}$ by independence. \square

We finally describe our quantum streaming algorithm for estimating the frequency moments. We trade off the number P of passes against the memory size of the algorithm by using the quantum sub-Gaussian estimator (Theorem 4.4.2) on a q-random variable derived from Proposition 9.4.2. The algorithm uses Proposition 9.3.5 and Proposition 9.3.3 to perform one quantum experiment at each pass on the input stream.

Theorem 9.4.3 (FREQUENCY MOMENT ESTIMATION). *For any integer $P \geq 1$, there exists a P -pass streaming algorithm with the following properties. Given an input stream in the turnstile model with non-zero final vector $x \in \mathbb{Z}^n$, an integer $k \geq 3$ and a real $\epsilon \in (\frac{10}{n}, 1)$, the algorithm uses a quantum memory of size $\tilde{O}\left(\frac{n^{1-2/k}}{(\epsilon P)^2}\right)$ and it outputs an estimate \tilde{F}_k such that $|\tilde{F}_k - F_k(x)| \leq \epsilon F_k(x)$ with probability at least $3/4$.*

Proof. We can assume in the proof that P is larger than some universal constant. If it is not the case, then we can instead use any optimal one-pass classical algorithm (such as [LW13; Gan15]) for estimating $F_k(x)$ directly with memory size $\tilde{O}(n^{1-2/k})$.

Our algorithm consists of first making one pass over the stream to compute an estimate \tilde{F}_2 such that $|\tilde{F}_2 - F_2(x)| \leq \frac{\epsilon}{16} F_2(x)$ with success probability at least $9/10$ by using a classical streaming algorithm [AMS99; KNW10] with memory size $O(\log(n)/\epsilon^2)$. For the subsequent passes, let us set $C = \frac{n^{1-2/k}}{(\epsilon P)^2} d^2 \log^4(P)$ for a sufficiently large constant d that will be defined later on. We apply Proposition 9.3.5 and Proposition 9.3.3 to the single-pass linear sketch algorithm presented in Proposition 9.4.2 with parameters $C, k, \epsilon/2, \tilde{F}_2$. We obtain that there exist a q-random variable F and two quantum streaming algorithms \mathcal{Q} and $\overline{\mathcal{Q}}$ with memory size $\tilde{O}\left(\frac{n^{1-2/k}}{(\epsilon P)^2}\right)$ such that $|\mathbb{E}[F] - F_k(x)| \leq \frac{\epsilon}{2} F_k(x)$, $\text{Var}[F] \leq \frac{2(\epsilon P F_k(x))^2}{d^2 \log^4(P)}$ and one quantum experiment with respect to F can be performed by one pass of \mathcal{Q} or $\overline{\mathcal{Q}}$ on the input stream. Consequently, we can construct a quantum streaming algorithm that runs the quantum sub-Gaussian estimator $\text{SubGaussEst}(F, t, 1/10)$ with parameter $t = P \frac{2\sqrt{2} \log(10)}{d \log^2(P)}$ by doing $d't \log^{3/2}(t) \log \log(t)$ passes over the stream for some universal constant d' (Theorem 4.4.2). We choose d such that $d't \log^{3/2}(t) \log \log(t) \leq P - 1$. The estimate \tilde{F}_k returned by this estimator satisfies,

$$|\tilde{F}_k - F_k(x)| \leq |\tilde{F}_k - \mathbb{E}[F]| + |\mathbb{E}[F] - F_k(x)| \leq \frac{\sqrt{\text{Var}[F]} \log(10)}{t} + \frac{\epsilon}{2} F_k(x) \leq \epsilon F_k(x)$$

with probability at least $9/10$, where the first inequality is the triangle inequality and the second one is by Theorem 4.4.2. The memory size used by the quantum sub-Gaussian estimator (Algorithm 4.2) is of the same order of magnitude as that used by \mathcal{Q} and $\overline{\mathcal{Q}}$. \square

9.5 Discussion

We conjecture that any P -pass quantum streaming algorithm for approximating the frequency moment $F_k(x)$ must use a memory of size $S \geq \Omega\left(\frac{n^{1-2/k}}{P^2}\right)$, meaning that the algorithm of Theorem 9.4.3 is (nearly) optimal. One way to answer this problem is to improve the lower bounds given by Jain, Radhakrishnan and Sen [JRS03] for the \mathcal{L}_∞ or the multi-party Set Disjointness (see the discussion in Section 9.1.1). In the \mathcal{L}_∞ problem, Alice and Bob are given two vectors $X, Y \in \{0, 1, \dots, m\}^n$ with the promise that either $|X_i - Y_i| \leq 1$ for all $i \in [n]$, or there exists an $i \in [n]$ such that $|X_i - Y_i| = m$. [JRS03] showed that the two-party r -round quantum communication complexity of \mathcal{L}_∞ is $\Omega(n/(r^3 m^{r+1}))$. It may be possible to improve this lower bound to $\Omega(n/(r m^2) + r)$ by using the techniques developed in [BGK+18], which would solve our conjecture.

10

Time-Space Tradeoffs by Recording Queries

This chapter is based on the following paper:

[HM21b] Y. Hamoudi and F. Magniez. “Quantum Time-Space Tradeoff for Finding Multiple Collision Pairs”. In: *Proceedings of the 16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*. 2021.

10.1 Introduction

Time-space tradeoffs aim at connecting the study of the *time* complexity, measured in this chapter as the number T of queries to an oracle, and the *space* complexity, which is the memory size S needed to execute an algorithm. The development of quantum computing asks the question of how the access to quantum operations and quantum memories modifies the tradeoff between these two computational resources. In some cases [BHT98b; Amb07; LZ19a], the only known ways to obtain a quantum speedup (in terms of time complexity) come at the cost of a dramatic increase in the space requirement. This raises the question to what extent can we combine time and space efficiency in the quantum setting. In this chapter, we approach this problem from the perspective of time-space tradeoff *lower bounds*, where the goal is to quantify how much the time to solve a given problem *must* increase when the available space decreases.

Our main object of study is the problem of finding K collision pairs in a uniformly random function $f : [N] \rightarrow [N]$. This task plays a central role in cryptography with the meet-in-the-middle type of attacks [OW99; Din20; Wag02; JL09; DDKS12; DEM19]. The celebrated algorithm of Brassard, Høyer and Tapp [BHT98b] paved the way to quantum speedups for solving this problem and its variants [CNS17; LZ19a; HSTX20]. Nevertheless, the space usage of these algorithms is much larger than that of the classical Pollard’s rho method [Pol75; OW99]. A time-space tradeoff lower bound is of great interest here to assess the security level of cryptographic schemes that are prone to collision attacks [Wie04; Ber05; Ber09]. We give the first evidence that the time complexity required to find K collision pairs must increase when the size of the available quantum memory is limited. More precisely, we show that any quantum algorithm using S qubits of memory must perform a number T of queries that satisfies the tradeoff $TS^{1/3} \geq \Omega(KN^{1/3})$ for finding K collision pairs. This represents a gap of at most $\tilde{\Omega}(K^{1/3})$ (when $S = O(\log N)$) with the optimal query complexity $T = \Theta(K^{2/3}N^{1/3})$ in the unlimited memory setting.

The main technique for studying time-space tradeoffs is the time-segmentation method [BFK+81; KŠW07] that converts a time lower bound proved in the *exponentially small success probability regime* into a time-space tradeoff lower bound. We develop a new approach for addressing the small success probability regime, based on a novel application

of Zhandry’s recording query technique [Zha19]. This technique consists of keeping track of the queries made by an algorithm to show that a significant portion of the input must be guessed randomly at the end of the computation. The “recording” of quantum queries is a subtle task that requires not to disturb the memory of the algorithm. We simplify the approach of Zhandry for doing that, and we generalize it to a broader class of problems. As a first application, we give a precise characterization of the best possible success probability for finding K collision pairs by using T queries, which leads to our time-space tradeoff. As a second application, we consider the K -Search problem on a random function $f : [N] \rightarrow \{0, 1\}$ where $f(x) = 1$ with probability K/N for each x . We apply the above machinery to give a simple analysis of the best possible success probability for solving this problem, and we use our result to reprove a time-space tradeoff for the Sorting problem.

10.1.1 Related work

Time-space tradeoffs have been studied for a long time in the classical branching program model [BFK+81; Bea91; BFM+87; Yao94; BSSV03; Abr90; MNT93]. The few results known in the quantum circuit model with oracle gates are for the Sorting problem [KŠW07], Boolean Matrix-Vector and Matrix-Matrix Multiplication [KŠW07] and Evaluating Solutions to Systems of Linear Inequalities [AŠW09]. A different line of works has studied the “Quantum Random Oracle Model with Auxiliary Input” [NABT15; HXY19; CLQ20; CGLQ20], where the memory restriction is enforced only on the size S of a precomputed advice provided to the algorithm. The recording technique has been used in [CGLQ20] to improve several lower bounds in this model. Apart from our work, all existing quantum tradeoffs are based on the hardness of Quantum Search.

The quantum recording query technique was invented by Zhandry [Zha19] to give new security proofs in the quantum random oracle model. It has been used to this end by many authors [HI19; LZ19b; CMS19; CMSZ19; BHH+19; AMRS20]. Zhandry has also illustrated its value for proving quantum query complexity lower bounds, by considering the Grover Search, Collision and k -Sum problems [Zha19]. Later, Liu and Zhandry [LZ19a] have applied it to the Multi-Collisions Finding problem. The two other main techniques for proving lower bounds in the quantum query model are the polynomial [BBC+01] and the adversary [Amb02] methods. Both methods gave results for the exponentially small success probability regime of the K -Search problem [KŠW07; Amb10a; Špa08] and in Strong Direct Product Theorems (SDPT) [Amb10a; AŠW09; Špa08; AMRR11; LR13]. An SDPT states that the success probability for solving K instances of a problem with less than K times the resources needed for one instance is exponentially small in K .

The classical Parallel Collision Search algorithm [OW99] (which generalizes the Pollard’s rho method [Pol75]) can find K collision pairs with a time-space tradeoff of $T^2S = \tilde{O}(K^2N)$ for any amount S of memory between $\tilde{\Omega}(\log N)$ and $\tilde{O}(K)$. The optimality of this tradeoff has been shown recently by Chakrabarti and Chen [CC17] (for 2-to-1 random functions) and by Dinur [Din20] (for uniformly random functions). The quantum BHT algorithm [BHT98b] can find a *single* collision pair in time $T = \tilde{O}(N^{1/3})$ and space $S = \tilde{O}(N^{1/3})$, which is optimal with respect to the time complexity [AS04; Zha15]. In comparison, the Pollard’s rho method uses $T = \tilde{O}(\sqrt{N})$ and $S = \tilde{O}(\log N)$. The question of whether a better quantum time-space tradeoff exists when $K = 1$ is a long-standing open question, that we do not settle here. The related *Element Distinctness* problem asks to find a collision pair in a random function $f : [N] \rightarrow [N^2]$ (or, more generally, to decide if a function contains a collision pair). There is some progress in the classical case [BFM+87; Yao94; BSSV03] toward proving an optimal time-space tradeoff for this problem.

10.1.2 Contributions and organization

We recall the quantum query model in Section 10.2.1 and we describe the space-bounded model used in this chapter in Section 10.2.2. The recording query framework is presented in Section 10.3. Our formalism is based on a general *recording query operator* (Definition 10.3.1) that can replace the standard quantum query operator without being noticed by the algorithm (Theorem 10.3.3). Next, we apply this framework to the Collision Pairs Finding problem defined as follows.

Definition 10.1.1. The *Collision Pairs Finding* problem asks to find a certain number K of disjoint collision pairs in a random function $f : [M] \rightarrow [N]$ where $M \geq N$. A *collision pair* (or simply *collision*) is a pair of values $x_1 \neq x_2$ such that $f(x_1) = f(x_2)$. Two collisions (x_1, x_2) and (x_3, x_4) are *disjoint* if x_1, \dots, x_4 are all different.

The requirement for the collisions to be disjoint is made to simplify our proofs later on. We note that a random function $f : [N] \rightarrow [N]$ contains $(1 - 2/e)N$ disjoint collisions on average [FO89]. Our first main result is to show that the optimal success probability decreases at an exponential rate in K when the number T of queries is smaller than $O(K^{2/3}N^{1/3})$. We note that, similarly to [Zha15] for the case $K = 1$, this bound is independent of the size M of the domain.

Theorem 10.4.6 (Restated). *The success probability of finding K disjoint collisions in a random function $f : [M] \rightarrow [N]$ is at most $O(T^3/(K^2N))^{K/2} + 2^{-K}$ for any algorithm making T quantum queries to f and any $1 \leq K \leq N/8$.*

Our second main result is the next time-space tradeoff lower bound for the same problem of finding K collision pairs. As a simple corollary, we obtain that finding almost all collisions using a memoryless algorithm (i.e. $S = O(\log N)$) requires to use at least $T \geq \Omega(N^{4/3})$ quantum queries, whereas $T = N$ classical queries are sufficient when there is no space restriction.

Theorem 10.6.1 (Restated). *Any quantum algorithm for finding K disjoint collisions in a random function $f : [M] \rightarrow [N]$ with success probability $2/3$ must satisfy a time-space tradeoff of $T^3S \geq \Omega(K^3N)$, where $1 \leq K \leq N/8$.*

We show in Algorithm 10.2 how to find an arbitrary number K of collisions with a tradeoff of $T^2S \leq \tilde{O}(K^2N)$. For the sake of simplicity in the analysis, we do not require these collisions to be disjoint. This is the same tradeoff as classically with the Parallel Collision Search algorithm [OW99], except that the space parameter S can hold larger values up to $\tilde{O}(K^{2/3}N^{1/3})$, hence the existence of a quantum speedup that matches the lower bound of Theorem 10.4.6 when there is no memory constraint.

Proposition 10.6.7 (Restated). *For any integers $1 \leq K \leq O(N)$ and $\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K^{2/3}N^{1/3})$, there exists a bounded-error quantum algorithm that can find K collisions in a random function $f : [N] \rightarrow [N]$ by making $T = \tilde{O}(K\sqrt{N/S})$ queries and using S qubits of memory.*

We summarize the classical and quantum time-space tradeoffs known for the Collision Pairs Finding problem in Table 10.1. We note that the tradeoff $T^2S \geq \Omega(K^2N)$ is always stronger than $T^3S \geq \Omega(K^3N)$ since $T \geq K$. We further show that any improvement to our time-space tradeoff lower bound would imply a breakthrough for the *Element Distinctness* problem, by showing the next reduction.

	Classical complexity	Quantum complexity
Upper bound:	$T^2S \leq \tilde{O}(K^2N)$ when $\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K)$ Parallel Collision Search [OW99]	$T^2S \leq \tilde{O}(K^2N)$ when $\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K^{2/3}N^{1/3})$ Proposition 10.6.7
Lower bound:	$T^2S \geq \Omega(K^2N)$ [Din20]	$T^3S \geq \Omega(K^3N)$ Theorem 10.6.1

Table 10.1: Complexity to find K disjoint collisions in a random function $f : [M] \rightarrow [N]$.

Corollary 10.6.5 (Restated). *Suppose that there exists $\epsilon \in (0, 1)$ such that any quantum algorithm for finding $\tilde{\Omega}(N)$ disjoint collisions in a random function $f : [10N] \rightarrow [N]$ must satisfy a time-space tradeoff of $TS^{1/3} \geq \tilde{\Omega}(N^{4/3+\epsilon})$. Then, any quantum algorithm for solving Element Distinctness on domain size N must satisfy a time-space tradeoff of $TS^{1/3} \geq \tilde{\Omega}(N^{2/3+2\epsilon})$.*

We point out that $TS^{1/3} \geq \Omega(N^{2/3})$ can already be deduced from the query complexity of Element Distinctness [AS04]. We conjecture that our current tradeoff for finding K collisions can be improved to $T^2S \geq \Omega(K^2N)$, which would imply $T^2S \geq \tilde{\Omega}(N^2)$ for Element Distinctness (Corollary 10.6.6). This result would be optimal [Amb07].

Finally, we adapt the machinery developed in this chapter to study the K -Search problem, which consists of finding K preimages of 1 in a random function $f : [M] \rightarrow \{0, 1\}$ where $f(x) = 1$ with probability K/N independently for each $x \in [M]$.

Theorem 10.5.1 (Restated). *The success probability of finding $K \leq N/8$ preimages of 1 in a random function $f : [M] \rightarrow \{0, 1\}$ where $f(x) = 1$ with probability K/N for each $x \in [M]$ is at most $O(T^2/(KN))^{K/2} + 2^{-K}$ for any algorithm using T quantum queries to f .*

As an application, we reprove in Theorem 10.6.8 the quantum time-space tradeoff for sorting N numbers first obtained in [KŠW07].

Theorem 10.6.8 (Restated). *Any quantum algorithm for sorting a function $f : [N] \rightarrow \{0, 1, 2\}$ with success probability $2/3$ must satisfy a time-space tradeoff of $T^2S \geq \Omega(N^3)$.*

10.1.3 Proof overview

Recording query technique. We use the recording query framework of Zhandry [Zha19] to upper bound the success probability of a query-bounded algorithm in finding K collision pairs. This method intends to reproduce the classical strategy where the queries made by an algorithm are recorded and answered with on-the-fly simulation of the oracle. Zhandry brought this technique to the quantum random oracle model by showing that, for the uniform input distribution, one can record *in superposition* the queries made by a quantum algorithm. Our first technical contribution (Section 10.3) is to simplify the analysis of Zhandry’s technique and, as a byproduct, to generalize it to any product distribution on the input. We notice that there has been other independent work on extending Zhandry’s recording technique [HI19; CMSZ19; CMS19]. Our approach is based on defining a “recording query operator” that is specific to the distribution under consideration. This operator can replace the standard quantum query operator without changing the success

probability of the algorithm, but with the effect of “recording” the quantum queries in an additional register. We detail two recording query operators corresponding to the uniform distribution (Lemma 10.4.2) and to the product of Bernoulli distributions (Lemma 10.5.3).

Finding collisions with time-bounded algorithms. Our application of the recording technique to the Collision Pairs Finding problem has two stages. We first bound the probability that the algorithm has forced the recording of many collisions after T queries. Namely, we show that the norm of the quantum state that records a new collision at the t -th query is on the order of $\sqrt{t/N}$ (Proposition 10.4.4). This is related to the probability that a new random value collides with one of the at most t previously recorded queries. The reason why the collisions have to be disjoint is to avoid the recording of more than one new collision in one query. By solving a simple recurrence relation, one gets that the amplitude of the basis states that have recorded at least $K/2$ collisions after T queries is at most $O(T^{3/2}/(K\sqrt{N}))^{K/2}$. We note that Liu and Zhandry [LZ19a, Theorem 5] carried out a similar analysis for the Multi-Collision Finding problem, where they obtained a similar bound of $O(T^{3/2}/\sqrt{N})^{K/2}$. The second stage of our proof relates the probability of having recorded many collisions to the actual success probability of the algorithm. If we used previous approaches (notably [Zha19, Lemma 5]), this step would degrade the upper bound on the success probability by adding a term that is polynomial in K/N . We preserve the exponentially small dependence on K by doing a more careful analysis of the relation between the recording and the standard query models (Proposition 10.4.5). We adopt a similar approach for the K -Search problem in Section 10.5.

Finding collisions with time-space bounded algorithms. We convert the above time-only bound into a time-space tradeoff by using the time-segmentation method [BFK+81; KŠW07]. Given a quantum circuit that solves the Collision Pairs Finding problem in time T and space S , we slice it into $T/(S^{2/3}N^{1/3})$ consecutive subcircuits, each of them using $S^{2/3}N^{1/3}$ queries. If no slice can output more than $\Omega(S)$ collisions with high probability then there must be at least $\Omega(K/S)$ slices in total, thus proving the desired tradeoff. Our previous lower bound implies that it is impossible to find $\Omega(S)$ collisions with probability larger than 4^{-S} in time $S^{2/3}N^{1/3}$. We must take into account that the initial memory at the beginning of each slice carries out information from previous stages. As in previous work [Aar05; KŠW07], we can “eliminate” this memory by replacing it with the completely mixed state while decreasing the success probability by a factor of 2^{-S} . Thus, if a slice outputs $\Omega(S)$ collisions then it can be used to contradict the lower bound proved before.

Element Distinctness. We connect the Collision Pairs Finding and Element Distinctness problems by showing how to transform a low-space algorithm for the latter into one for the former (Proposition 10.6.3). If there exists a time- \bar{T} space- \bar{S} algorithm for Element Distinctness on domain size \sqrt{N} then we can find $\tilde{\Omega}(N)$ collisions in a random function $f : [N] \rightarrow [N]$ by repeatedly sampling a subset $H \subset [N]$ of size \sqrt{N} and using that algorithm on the function f restricted to the domain H . Among other things, we must ensure that the same collision does not occur many times and that storing the set H does not use too much memory (it turns out that 4-wise independence is sufficient for our purpose). We end up with an algorithm with time $T = O(N\bar{T})$ and space $S = O(\bar{S})$ for finding $\tilde{\Omega}(N)$ collisions. Consequently, if the Element Distinctness problem on domain size \sqrt{N} can be solved with a time-space tradeoff of $\bar{T}\bar{S}^{1/3} \leq O(N^{1/3+\epsilon})$, then there is an algorithm for finding $\tilde{\Omega}(N)$ collisions that satisfies a tradeoff of $TS^{1/3} \leq O(N^{4/3+\epsilon})$.

10.2 Models of computation

We first present the standard model of quantum query complexity in Section 10.2.1. This model is used for investigating the *time complexity* of the Collision Pairs Finding problem in Section 10.4, and of the K -Search problem in Section 10.5. Then, we describe the more general circuit model that also captures the *space complexity* in Section 10.2.2. It is used in Section 10.6 for studying time-space tradeoffs.

10.2.1 Query model

The (standard) model of quantum query complexity [BW02] measures the number of quantum queries an algorithm (also called an “attacker”) needs to make on an input function $f : [M] \rightarrow [N]$ to find an output z satisfying some predetermined relation $R(f, z)$. We present this model in more detail below.

Quantum Query Algorithm. A T -query quantum algorithm is specified by a sequence U_0, \dots, U_T of unitary transformations acting on the algorithm’s memory. The state $|\psi\rangle$ of the algorithm is made of three registers $\mathcal{Q}, \mathcal{P}, \mathcal{W}$ where the *query register* \mathcal{Q} holds $x \in [M]$, the *phase register* \mathcal{P} holds $p \in [N]$ and the *working register* \mathcal{W} holds some value w . We represent a basis state in the corresponding Hilbert space as $|x, p, w\rangle_{\mathcal{Q}\mathcal{P}\mathcal{W}}$. We may drop the subscript $\mathcal{Q}\mathcal{P}\mathcal{W}$ when it is clear from the context. The state $|\psi_t^f\rangle$ of the algorithm after $t \leq T$ queries to some input function $f : [M] \rightarrow [N]$ is

$$|\psi_t^f\rangle = U_t \mathcal{O}_f U_{t-1} \cdots U_1 \mathcal{O}_f U_0 |\mathbf{0}\rangle$$

where the oracle \mathcal{O}_f is defined by

$$\mathcal{O}_f |x, p, w\rangle = \omega_N^{pf(x)} |x, p, w\rangle \quad \text{and} \quad \omega_N = e^{\frac{2i\pi}{N}}.$$

The *output* of the algorithm is written on a substring z of the value w . The *success probability* σ_f of the quantum algorithm on f is the probability that the output value z obtained by measuring the working register of $|\psi_T^f\rangle$ in the computational basis satisfies the relation $R(f, z)$. In other words, if we let Π_{succ}^f be the projector whose support consists of all basis states $|x, p, w\rangle$ such that the output substring z of w satisfies $R(f, z)$, then $\sigma_f = \|\Pi_{\text{succ}}^f |\psi_T^f\rangle\|^2$.

Oracle’s Register. Here, we describe the variant used in the adversary method [Amb02] and in Zhandry’s work [Zha19]. It is represented as an interaction between an *algorithm* that aims at finding a correct output z , and a superposition of *oracle’s* inputs that respond to the queries from the algorithm.

The memory of the oracle is made of an *input register* \mathcal{F} holding the description of a function $f : [M] \rightarrow [N]$. This register is divided into M subregisters $\mathcal{F}_1, \dots, \mathcal{F}_M$ where \mathcal{F}_x holds $f(x) \in [N]$ for each $x \in [M]$. The basis states in the corresponding Hilbert space are $|f\rangle_{\mathcal{F}} = \otimes_{x \in [M]} |f(x)\rangle_{\mathcal{F}_x}$. Given an input distribution D on the set of functions $[N]^M$, the *oracle’s initial state* is the state $|\text{init}\rangle_{\mathcal{F}} = \sum_{f \in [N]^M} \sqrt{\Pr[f \leftarrow D]} |f\rangle$.

The *query operator* \mathcal{O} is a unitary transformation acting on the memory of the algorithm and the oracle. Its action is defined on each basis state by

$$\mathcal{O} |x, p, w\rangle |f\rangle = (\mathcal{O}_f |x, p, w\rangle) |f\rangle.$$

The joint state $|\psi_t\rangle$ of the algorithm and the oracle after t queries is equal to $|\psi_t\rangle = U_t \mathcal{O} U_{t-1} \cdots U_1 \mathcal{O} U_0 (|\mathbf{0}\rangle |\text{init}\rangle) = \sum_{f \in [N]^M} \sqrt{\Pr[f \leftarrow D]} |\psi_t^f\rangle |f\rangle$, where the unitaries U_i have

been extended to act as the identity on \mathcal{F} . The *success probability* σ of a quantum algorithm on an input distribution D is the probability that the output value z and the input f obtained by measuring the working and input registers of the final state $|\psi_T\rangle$ satisfy the relation $R(f, z)$. In other words, if we let Π_{succ} be the projector whose support consists of all basis states $|x, p, w\rangle|f\rangle$ such that the output substring z of w satisfies $R(f, z)$, then $\sigma = \|\Pi_{\text{succ}}|\psi_T\rangle\|^2$.

10.2.2 Space-bounded model

Our model of space-bounded computation is identical to the one described in [KŠW07; AŠW09]. We use the quantum circuit model (presented in Section 3.1) augmented with the oracle gates of the query model defined in the previous section. The *time complexity*, denoted by T , is the number of gates in the circuit. In practice, we lower bound it by the number of oracle gates only. The *space complexity*, denoted by S , is the number of qubits on which the circuit is operating. The result of the computation is written on some dedicated output qubits that may not be used later on, and that are *not counted toward the space bound*. In particular, the size of the output can be larger than S . Furthermore, we assume that the output qubits are updated at some predefined output gates in the circuit.

We notice that, by the deferred measurement principle, any space-bounded computation that uses T queries can be transformed into a T -query unitary algorithm as defined in Section 10.2.1. Thus, any lower bound on the query complexity of a problem is also a lower bound on the time complexity of that problem in the space-bounded model. This explains our use of the query model in Section 10.4 and Section 10.5.

10.3 Recording model

The quantum recording query model is a modification of the standard query model defined in Section 10.2.1 that is unnoticeable by the algorithm, but that allows us to track more easily the progress made toward solving the problem under consideration. The original recording model was formulated by Zhandry in [Zha19]. Here, we propose a simplified and more general version of this framework that only requires the initial oracle's state $|\text{init}\rangle_{\mathcal{F}}$ to be a product state $\otimes_{x \in [M]} |\text{init}_x\rangle_{\mathcal{F}_x}$ (instead of the uniform distribution over all basis states as in [Zha19]).

Construction. The range $[N]$ is augmented with a new symbol \perp . The input register \mathcal{F} of the oracle can now contain $f : [M] \rightarrow [N] \cup \{\perp\}$, where $f(x) = \perp$ represents the absence of knowledge from the algorithm about the image of x . Unlike in the standard query model, the oracle's initial state is independent of the input distribution and is fixed to be $|\perp^M\rangle_{\mathcal{F}}$ (which represents the fact that the algorithm knows nothing about the input initially). We extend the query operator \mathcal{O} defined in the standard query model by setting

$$\mathcal{O}|x, p, w\rangle|f\rangle = |x, p, w\rangle|f\rangle \quad \text{when } f(x) = \perp.$$

Given a product input distribution $D = D_1 \otimes \cdots \otimes D_M$ on the set $[N]^M$, the oracle's initial state in the standard query model can be decomposed as the product state $|\text{init}\rangle_{\mathcal{F}} = \otimes_{x \in [M]} |\text{init}_x\rangle_{\mathcal{F}_x}$ where $|\text{init}_x\rangle_{\mathcal{F}_x} := \sum_{y \in [N]} \sqrt{\Pr[y \leftarrow D_x]} |y\rangle_{\mathcal{F}_x}$. The “recording query operator” \mathcal{R} is defined with respect to a family $(\mathcal{S}_x)_{x \in [M]}$ of unitary operators satisfying $\mathcal{S}_x|\perp\rangle_{\mathcal{F}_x} = |\text{init}_x\rangle_{\mathcal{F}_x}$ for all x as follows.

Definition 10.3.1. Given M unitary operators $\mathcal{S}_1, \dots, \mathcal{S}_M$ acting on $\mathcal{F}_1, \dots, \mathcal{F}_M$ respectively, consider the operator \mathcal{S} acting on all the registers $\mathcal{QPW}\mathcal{F}$ such that,

$$\mathcal{S} = \sum_{x \in [M]} |x\rangle\langle x|_{\mathcal{Q}} \otimes I_{\mathcal{PW}\mathcal{F}_1 \dots \mathcal{F}_{x-1}} \otimes \mathcal{S}_x \otimes I_{\mathcal{F}_{x+1} \dots \mathcal{F}_M}.$$

Then, the *recording query operator* \mathcal{R} with respect to $(\mathcal{S}_x)_{x \in [M]}$ is defined as $\mathcal{R} = \mathcal{S}^\dagger \mathcal{O} \mathcal{S}$.

Later in this chapter, we describe two recording query operators related to the uniform distribution (Lemma 10.4.2) and to the product of Bernoulli distributions (Lemma 10.5.3).

Indistinguishability. The joint state of the algorithm and the oracle after t queries in the recording query model is defined as $|\phi_t\rangle = U_t \mathcal{R} U_{t-1} \dots U_1 \mathcal{R} U_0(|\mathbf{0}\rangle|\perp^M\rangle)$. Notice that the query operator \mathcal{R} can only change the value of $f(x')$ (contained in the register $\mathcal{F}_{x'}$) when it is applied to a state $|x, p, w\rangle|f\rangle$ such that $x = x'$. As a result, we have the following simple fact.

Fact 10.3.2. *The state $|\phi_t\rangle$ is a linear combination of basis states $|x, p, w\rangle|f\rangle$ where f contains at most t entries different from \perp .*

The entries of f that are different from \perp represent what the oracle has learned (or “recorded”) from the algorithm’s queries so far. In the next theorem, we show that $|\phi_t\rangle$ is related to the state $|\psi_t\rangle$ (defined in Section 10.2.1) by $|\psi_t\rangle = (I_{\mathcal{QPW}} \otimes_{x \in [M]} \mathcal{S}_x)|\phi_t\rangle$. In particular, the states $|\psi_t\rangle$ and $|\phi_t\rangle$ cannot be distinguished by the algorithm since the reduced states on the algorithm’s registers are identical.

Theorem 10.3.3. *Let (U_0, \dots, U_T) be a T -query quantum algorithm. Given M unitary operators $\mathcal{S}_1, \dots, \mathcal{S}_M$ acting on the oracle’s registers $\mathcal{F}_1, \dots, \mathcal{F}_M$ respectively, let \mathcal{R} denote the recording query operator associated with $(\mathcal{S}_x)_{x \in [M]}$, and define the initial state $|\text{init}\rangle_{\mathcal{F}} = (\otimes_{x \in [M]} \mathcal{S}_x)|\perp^M\rangle$. Then, the states*

$$\begin{cases} |\psi_t\rangle = U_t \mathcal{O} U_{t-1} \dots U_1 \mathcal{O} U_0(|\mathbf{0}\rangle|\text{init}\rangle) \\ |\phi_t\rangle = U_t \mathcal{R} U_{t-1} \dots U_1 \mathcal{R} U_0(|\mathbf{0}\rangle|\perp^M\rangle) \end{cases}$$

after $t \leq T$ queries in the standard and recording query models respectively satisfy

$$|\psi_t\rangle = \mathcal{T}|\phi_t\rangle \quad \text{where} \quad \mathcal{T} = I_{\mathcal{QPW}} \bigotimes_{x \in [M]} \mathcal{S}_x.$$

Proof. We start by introducing the intermediate operator $\bar{\mathcal{R}} = \mathcal{T}^\dagger \mathcal{O} \mathcal{T}$. Observe that for any basis state $|x, p, w\rangle|f\rangle$ the operators $\bar{\mathcal{R}}$ and \mathcal{R} act the same way on the registers \mathcal{QPF}_x and they do not depend on the other registers. Thus, we have $\bar{\mathcal{R}} = \mathcal{R}$. We further observe that U_i and \mathcal{T} commute for all i since they depend on disjoint registers. Consequently, we have that

$$\begin{aligned} |\psi_t\rangle &= U_t \mathcal{O} U_{t-1} \mathcal{O} \dots U_1 \mathcal{O} U_0 \cdot \mathcal{T}(|\mathbf{0}\rangle|\perp^M\rangle) && \text{since } \mathcal{T}(|\mathbf{0}\rangle|\perp^M\rangle) = |\mathbf{0}\rangle|\text{init}\rangle \\ &= \mathcal{T} \mathcal{T}^\dagger U_t \mathcal{O} \cdot \mathcal{T} \mathcal{T}^\dagger U_{t-1} \mathcal{O} \dots \mathcal{T} \mathcal{T}^\dagger U_1 \mathcal{O} \cdot \mathcal{T} \mathcal{T}^\dagger U_0 \cdot \mathcal{T}(|\mathbf{0}\rangle|\perp^M\rangle) && \text{since } \mathcal{T} \mathcal{T}^\dagger = I \\ &= \mathcal{T} U_t \mathcal{T}^\dagger \cdot \mathcal{O} \cdot \mathcal{T} U_{t-1} \mathcal{T}^\dagger \cdot \mathcal{O} \dots \mathcal{T} U_1 \mathcal{T}^\dagger \cdot \mathcal{O} \cdot \mathcal{T} U_0(|\mathbf{0}\rangle|\perp^M\rangle) && \text{by commutation} \\ &= \mathcal{T} U_t \bar{\mathcal{R}} U_{t-1} \dots U_1 \bar{\mathcal{R}} U_0(|\mathbf{0}\rangle|\perp^M\rangle) && \text{by definition of } \bar{\mathcal{R}} \\ &= \mathcal{T} U_t \mathcal{R} U_{t-1} \dots U_1 \mathcal{R} U_0(|\mathbf{0}\rangle|\perp^M\rangle) && \text{since } \bar{\mathcal{R}} = \mathcal{R} \\ &= \mathcal{T}|\phi_t\rangle && \text{by definition of } |\phi_t\rangle. \end{aligned}$$

□

10.4 Time lower bound for Collision Pairs Finding

In this section, we upper bound the success probability of finding K disjoint collisions in the query-bounded model of Section 10.2.1. The proof uses the recording query model of Section 10.3. We first describe in Section 10.4.1 the recording query framework associated with this problem. In Section 10.4.2, we study the probability that an algorithm has recorded at least $k \leq K$ collisions after $t \leq T$ queries. We prove by induction on t and k that this quantity is exponentially small in k when $t \leq O(k^{2/3}N^{1/3})$ (Proposition 10.4.4). Finally, in Section 10.4.3, we relate this progress measure to the actual success probability (Proposition 10.4.5), and we conclude that the latter quantity is exponentially small in K after $T \leq O(K^{2/3}N^{1/3})$ queries (Theorem 10.4.6).

10.4.1 Recording query operator

We describe a recording operator that corresponds to the uniform distribution on the set of functions $f : [M] \rightarrow [N]$. In the standard query model, the oracle's initial state is $|\text{init}\rangle_{\mathcal{F}} = \otimes_{x \in [M]} \left(\frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle_{\mathcal{F}_x} \right)$. Consequently, in the recording query model, we choose the unitary transformations $\mathcal{S}_1, \dots, \mathcal{S}_M$ to be defined as follows.

Definition 10.4.1. For any $x \in [M]$, we define the unitary \mathcal{S}_x acting on the register \mathcal{F}_x to be

$$\mathcal{S}_x : \begin{cases} |\perp\rangle_{\mathcal{F}_x} & \mapsto \frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle_{\mathcal{F}_x} \\ \frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle_{\mathcal{F}_x} & \mapsto |\perp\rangle_{\mathcal{F}_x} \\ \frac{1}{\sqrt{N}} \sum_{y \in [N]} \omega_N^{py} |y\rangle_{\mathcal{F}_x} & \mapsto \frac{1}{\sqrt{N}} \sum_{y \in [N]} \omega_N^{py} |y\rangle_{\mathcal{F}_x} \quad \text{for } p = 1, \dots, N-1. \end{cases}$$

These unitaries verify $\mathcal{T}|\perp^M\rangle = |\text{init}\rangle$ where $\mathcal{T} = \otimes_{x \in [M]} \mathcal{S}_x$, as required by Theorem 10.3.3. The recording query operator is $\mathcal{R} = \mathcal{S}\mathcal{O}\mathcal{S}$ (Definition 10.3.1) since $\mathcal{S}^\dagger = \mathcal{S}$. The next lemma gives an explicit characterization of the action of \mathcal{R} on a basis state.

Lemma 10.4.2. *If the recording query operator \mathcal{R} associated with Definition 10.4.1 is applied to a basis state $|x, p, w\rangle|f\rangle$ where $p \neq 0$ then the register $|f(x)\rangle_{\mathcal{F}_x}$ is mapped to*

$$\begin{cases} \sum_{y \in [N]} \frac{\omega_N^{py}}{\sqrt{N}} |y\rangle & \text{if } f(x) = \perp \\ \frac{\omega_N^{pf(x)}}{N} |\perp\rangle + \frac{1 + \omega_N^{pf(x)}(N-2)}{N} |f(x)\rangle + \sum_{y \in [N] \setminus \{f(x)\}} \frac{1 - \omega_N^{py} - \omega_N^{pf(x)}}{N} |y\rangle & \text{otherwise} \end{cases}$$

and the other registers are unchanged. If $p = 0$ then none of the registers are changed.

Proof. By definition, the unitary \mathcal{S}_x maps $|\perp\rangle_{\mathcal{F}_x} \mapsto \frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle$ and $|y\rangle_{\mathcal{F}_x} \mapsto \frac{1}{\sqrt{N}} |\perp\rangle + \frac{1}{\sqrt{N}} \sum_{p' \in [N] \setminus \{0\}} \omega_N^{-p'y} |\widehat{p'}\rangle$ where $y \in [N]$ and $|\widehat{p'}\rangle := \frac{1}{\sqrt{N}} \sum_{y \in [N]} \omega_N^{p'y} |y\rangle$. Thus, the action on the register \mathcal{F}_x is:

- If $f(x) = \perp$ then $|f(x)\rangle_{\mathcal{F}_x} \xrightarrow{\mathcal{S}} \frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle \xrightarrow{\mathcal{O}} \frac{1}{\sqrt{N}} \sum_{y \in [N]} \omega_N^{py} |y\rangle \xrightarrow{\mathcal{S}} \frac{1}{\sqrt{N}} \sum_{y \in [N]} \omega_N^{py} |y\rangle$.
- If $f(x) \in [N]$ then $|f(x)\rangle_{\mathcal{F}_x} = \frac{1}{\sqrt{N}} \sum_{p' \in [N]} \omega_N^{-p'f(x)} |\widehat{p'}\rangle \xrightarrow{\mathcal{S}} \frac{1}{\sqrt{N}} |\perp\rangle + \frac{1}{\sqrt{N}} \sum_{p' \in [N] \setminus \{0\}} \omega_N^{-p'f(x)} |\widehat{p'}\rangle \xrightarrow{\mathcal{O}} \frac{1}{\sqrt{N}} |\perp\rangle + \frac{1}{\sqrt{N}} \sum_{p' \in [N] \setminus \{0\}} \omega_N^{-p'f(x)} |\widehat{p' + p}\rangle = \frac{1}{\sqrt{N}} |\perp\rangle + \frac{\omega_N^{pf(x)}}{\sqrt{N}} \sum_{p' \in [N] \setminus \{p\}} \omega_N^{-p'f(x)} |\widehat{p'}\rangle$

$$\begin{aligned} \xrightarrow{\mathcal{S}} & \frac{1}{N} \sum_{y \in [N]} |y\rangle + \frac{\omega_N^{pf(x)}}{\sqrt{N}} |\perp\rangle + \frac{\omega_N^{pf(x)}}{\sqrt{N}} \sum_{p' \in [N] \setminus \{0, p\}} \omega_N^{-p'f(x)} |\widehat{p'}\rangle = \frac{\omega_N^{pf(x)}}{N} |\perp\rangle + \frac{1 + \omega_N^{pf(x)}(N-2)}{N} \\ & |f(x)\rangle + \sum_{y \in [N] \setminus \{f(x)\}} \frac{1 - \omega_N^{py} - \omega_N^{pf(x)}}{N} |y\rangle. \end{aligned}$$

□

We note that the recording operator \mathcal{R} is close to the mapping $|\perp\rangle_{\mathcal{F}_x} \mapsto \sum_{y \in [N]} \frac{\omega_N^{py}}{\sqrt{N}} |y\rangle$ and $|f(x)\rangle_{\mathcal{F}_x} \mapsto \omega_N^{pf(x)} |f(x)\rangle$ (if $f(x) \neq \perp$) up to lower-order terms of amplitude $O(1/N)$. This is analogous to a “lazy” classical oracle that would choose the value of $f(x)$ uniformly at random the first time it is queried.

10.4.2 Analysis of the recording progress

We define a measure of progress based on the number of disjoint collisions contained in the oracle’s register of the recording query model. We first give some projectors related to this quantity.

Definition 10.4.3. We define the following projectors by giving the basis states on which they project:

- $\Pi_{\leq k}$, $\Pi_{=k}$ and $\Pi_{\geq k}$: all basis states $|x, p, w\rangle|f\rangle$ such that f contains respectively *at most*, *exactly* or *at least* k disjoint collisions (the entries with \perp are not considered as collisions).
- $\Pi_{=k, \perp}$ and $\Pi_{=k, y}$ for $y \in [N]$: all basis states $|x, p, w\rangle|f\rangle$ such that (1) f contains *exactly* k disjoint collisions, (2) the phase multiplier p is nonzero and (3) $f(x) = \perp$ or $f(x) = y$ respectively.

We can now define the measure of progress $q_{t,k}$ for t queries and k collisions as

$$q_{t,k} = \|\Pi_{\geq k}|\phi_t\rangle\|$$

where $|\phi_t\rangle$ is the state after t queries in the recording query model. The main result of this section is the following bound on the growth of $q_{t,k}$.

Proposition 10.4.4. *For all t and k , we have that $q_{t,k} \leq \binom{t}{k} \left(\frac{4\sqrt{t}}{\sqrt{N}}\right)^k$.*

Proof. First, $q_{0,0} = 1$ and $q_{0,k} = 0$ for all $k \geq 1$ since the initial state is $|\phi_0\rangle = |\mathbf{0}\rangle|\perp\rangle^M$. Then, we prove that $q_{t,k}$ satisfies the following recurrence relation

$$q_{t+1,k+1} \leq q_{t,k+1} + 4\sqrt{\frac{t}{N}} q_{t,k}. \quad (10.1)$$

From this result, it is trivial to conclude that $q_{t,k} \leq \binom{t}{k} \left(\frac{4\sqrt{t}}{\sqrt{N}}\right)^k$. In order to prove Equation (10.1), we first observe that $q_{t+1,k+1} = \|\Pi_{\geq k+1} U_{t+1} \mathcal{R} |\phi_t\rangle\| = \|\Pi_{\geq k+1} \mathcal{R} |\phi_t\rangle\|$ since the unitary U_{t+1} applied by the algorithm at time $t+1$ does not modify the oracle’s memory. Then, on any basis state $|x, p, w\rangle|f\rangle$, the recording query operator \mathcal{R} acts as the identity on the registers $\mathcal{F}_{x'}$ for $x' \neq x$. Consequently, the basis states $|x, p, w\rangle|f\rangle$ in $|\phi_t\rangle$ that may contribute to $q_{t+1,k+1}$ must either already contain $k+1$ disjoint collisions in f , or exactly k disjoint collisions in f and $p \neq 0$. This implies that

$$q_{t+1,k+1} \leq q_{t,k+1} + \|\Pi_{\geq k+1} \mathcal{R} \Pi_{=k, \perp} |\phi_t\rangle\| + \sum_{y \in [N]} \|\Pi_{\geq k+1} \mathcal{R} \Pi_{=k, y} |\phi_t\rangle\|.$$

We first bound the term $\|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,\perp}|\phi_t\rangle\|$. Consider any basis state $|x, p, w\rangle|f\rangle$ in the support of $\Pi_{=k,\perp}$ and $|\phi_t\rangle$. The function f must contain at most t entries different from \perp by Fact 10.3.2. By Lemma 10.4.2, we have $\mathcal{R}|x, p, w\rangle|f\rangle = \sum_{y \in [N]} \frac{\omega_N^{py}}{\sqrt{N}} |x, p, w\rangle|y\rangle_{\mathcal{F}_x} \otimes_{x' \neq x} |f(x')\rangle_{\mathcal{F}_{x'}}$. Since there are at most t entries in f that can collide with the value contained in the register \mathcal{F}_x , we have $\|\Pi_{\geq k+1}\mathcal{R}|x, p, w\rangle|f\rangle\| \leq \sqrt{t/N}$. Finally, since any two basis states in the support of $\Pi_{=k,\perp}$ remain orthogonal after $\Pi_{\geq k+1}\mathcal{R}$ is applied, we obtain that $\|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,\perp}|\phi_t\rangle\| \leq \sqrt{t/N}\|\Pi_{=k,\perp}|\phi_t\rangle\| \leq \sqrt{t/N}q_{t,k}$.

We now consider the term $\|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,y}|\phi_t\rangle\|$ for any $y \in [N]$. Again, we consider any basis state $|x, p, w\rangle|f\rangle$ in the support of $\Pi_{=k,y}$ where f has at most t entries different from \perp . According to Lemma 10.4.2, we have $\mathcal{R}|x, p, w\rangle|f\rangle = \frac{\omega_N^{pf(x)}}{N}|\perp\rangle + \frac{1+\omega_N^{pf(x)}(N-2)}{N}|f(x)\rangle + \sum_{y' \neq f(x)} \frac{1-\omega_N^{py'}-\omega_N^{pf(x)}}{N}|x, p, w\rangle|y'\rangle_{\mathcal{F}_x} \otimes_{x' \neq x} |f(x')\rangle_{\mathcal{F}_{x'}}$. As before, there are at most t terms in this sum that can be in the support of $\Pi_{\geq k+1}$. Consequently, $\|\Pi_{\geq k+1}\mathcal{R}|x, p, w\rangle|f\rangle\| \leq 3\sqrt{t/N}$ and $\|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,y}|\phi_t\rangle\| \leq 3\sqrt{t/N}\|\Pi_{=k,y}|\phi_t\rangle\|$.

We conclude that $q_{t+1,k+1} \leq q_{t,k+1} + \sqrt{t/N}q_{t,k} + \sum_{y \in [N]} 3\sqrt{t/N}\|\Pi_{=k,y}|\phi_t\rangle\| \leq q_{t,k+1} + \sqrt{t/N}q_{t,k} + 3\sqrt{t/N}\sqrt{\sum_{y \in [N]}\|\Pi_{=k,y}|\phi_t\rangle\|^2} \leq q_{t,k+1} + \sqrt{t/N}q_{t,k} + 3\sqrt{t/N}q_{t,k}$, where the second step is by Cauchy-Schwarz' inequality. \square

10.4.3 From the recording progress to the success probability

We connect the success probability $\sigma = \|\Pi_{\text{succ}}|\psi_T\rangle\|^2$ in the standard query model to the final progress $q_{T,k}$ in the recording query model after T queries. We show that if the algorithm has made no significant progress for recording $k \geq K/2$ collisions then it needs to “guess” the positions of $K - k$ other collisions. Classically, the probability to find the values of $K - k$ collisions that have not been queried would be at most $(1/N^2)^{K-k}$. Here, we show similarly that if a unit state contains at most k collisions in the quantum recording model, then after mapping it to the standard query model (by applying the operator \mathcal{T} of Theorem 10.3.3) the probability that the output register contains the correct positions of K collisions is at most $N^2(4K^2/N^2)^{K-k}$.

Proposition 10.4.5. *For any state $|\phi\rangle$, we have $\|\Pi_{\text{succ}}\mathcal{T}\Pi_{\leq k}|\phi\rangle\| \leq N\left(\frac{2K}{N}\right)^{K-k}\|\Pi_{\leq k}|\phi\rangle\|$.*

Proof. We assume that the output of the algorithm also contains the image of each collision pair under f . Namely, the output z is represented as a list of K triples $(x_1, x_2, y_1), \dots, (x_{2K-1}, x_{2K}, y_K) \in [M]^2 \times ([N] \cup \{\perp\})$. It is correct if the input function $f: [M] \rightarrow [N]$ (in the standard query model) satisfies $f(x_{2i-1}) = f(x_{2i}) = y_i \neq \perp$ for all $1 \leq i \leq K$, and the values x_1, x_2, \dots, x_{2K} are all different. By definition, the support of Π_{succ} consists of all basis states $|x, p, w\rangle|f\rangle$ such that the output substring z of w satisfies these conditions.

We define a new family of projectors $\tilde{\Pi}_{a,b}$, where $0 \leq a+b \leq 2K$, whose supports consist of all basis states $|x, p, w\rangle|f\rangle$ satisfying the following conditions:

- (A) The output substring z is made of K triples $(x_1, x_2, y_1), \dots, (x_{2K-1}, x_{2K}, y_K)$ where the x_i are all different.
- (B) There are exactly a indices $i \in [2K]$ such that $f(x_i) = \perp$.
- (C) There are exactly b indices $i \in [2K]$ such that $f(x_i) \neq \perp$ and $f(x_i) \neq y_{\lceil i/2 \rceil}$.

For any state $|x, p, w\rangle|f\rangle$ in the support of $\tilde{\Pi}_{a,b}$, we claim that

$$\|\Pi_{\text{succ}}\mathcal{T}|x, p, w\rangle|f\rangle\| \leq \left(\frac{1}{\sqrt{N}}\right)^a \left(\frac{1}{N}\right)^b. \quad (10.2)$$

Indeed, we have $\mathcal{T} = \otimes_{x' \in [M]} \mathcal{S}_{x'}$ and by Definition 10.4.1 the action of \mathcal{S}_{x_i} on the register $|f(x_i)\rangle_{\mathcal{F}_{x_i}}$ is $|f(x_i)\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle$ if $f(x_i) = \perp$, and $|f(x_i)\rangle \mapsto \frac{1}{\sqrt{N}} |\perp\rangle + (1 - \frac{1}{N}) |f(x_i)\rangle - \frac{1}{N} \sum_{y \in [N] \setminus \{f(x_i)\}} |y\rangle$ otherwise. The projector Π_{succ} only keeps the term $|y_{[i/2]}\rangle$ in these sums, which implies Equation (10.2).

Let us now consider any linear combination $|\varphi\rangle = \sum_{x,p,w,f} \alpha_{x,p,w,f} |x,p,w\rangle |f\rangle$ of basis states that are in the support of $\tilde{\Pi}_{a,b}$. We claim that

$$\|\Pi_{\text{succ}} \mathcal{T} |\varphi\rangle\| \leq \left(\sqrt{\frac{2K}{N}} \right)^{a+b} \|\varphi\rangle\|. \quad (10.3)$$

First, given two states $|x,p,w\rangle |f\rangle$ and $|\bar{x},\bar{p},\bar{w}\rangle |\bar{f}\rangle$ where $z = ((x_1, x_2, y_1), \dots, (x_{2K-1}, x_{2K}, y_K))$ is the output substring of w , if the tuples $(x, p, w, (f(x'))_{x' \notin \{x_1, \dots, x_{2K}\}})$ and $(\bar{x}, \bar{p}, \bar{w}, (\bar{f}(x'))_{x' \notin \{x_1, \dots, x_{2K}\}})$ are different then the state $\Pi_{\text{succ}} \mathcal{T} |x, p, w\rangle |f\rangle$ must be orthogonal to $\Pi_{\text{succ}} \mathcal{T} |\bar{x}, \bar{p}, \bar{w}\rangle |\bar{f}\rangle$. Moreover, for any $z = ((x_1, x_2, y_1), \dots, (x_{2K-1}, x_{2K}, y_K))$ that satisfies condition (A), there are $\binom{2K}{a} \binom{2K-a}{b} (N-1)^b \leq (2K)^{a+b} N^b$ different ways to choose $(f(x_i))_{i \in [2K]}$ that satisfy conditions (B) and (C). Let us write $w_{\bar{x}} = \{x_1, \dots, x_{2K}\}$ when the output substring z of w contains x_1, \dots, x_{2K} . Then, by using the Cauchy-Schwarz inequality and Equation (10.2), we get that

$$\begin{aligned} \|\Pi_{\text{succ}} \mathcal{T} |\varphi\rangle\|^2 &= \sum_{x,p,w, (f(x'))_{x' \notin w_{\bar{x}}}} \left\| \sum_{(f(x'))_{x' \in w_{\bar{x}}}} \alpha_{x,p,w,f} \Pi_{\text{succ}} \mathcal{T} |x, p, w\rangle |f\rangle \right\|^2 \\ &\leq \sum_{x,p,w, (f(x'))_{x' \notin w_{\bar{x}}}} \left(\sum_{(f(x'))_{x' \in w_{\bar{x}}}} |\alpha_{x,p,w,f}|^2 \right) \left(\sum_{(f(x'))_{x' \in w_{\bar{x}}}} \|\Pi_{\text{succ}} \mathcal{T} |x, p, w\rangle |f\rangle\|^2 \right) \\ &\leq \|\varphi\rangle\|^2 \cdot (2K)^{a+b} N^b \left(\frac{1}{N} \right)^a \left(\frac{1}{N^2} \right)^b \\ &= \left(\frac{2K}{N} \right)^{a+b} \|\varphi\rangle\|^2, \end{aligned}$$

which proves Equation (10.3). Observe now that the support of $\Pi_{\leq k}$ is contained into the union of the supports of $\tilde{\Pi}_{a,b}$ for $a+b \geq 2(K-k)$. Thus, by the triangle inequality, $\|\Pi_{\text{succ}} \mathcal{T} \Pi_{\leq k} |\phi\rangle\| \leq \sum_{a+b \geq 2(K-k)} \|\Pi_{\text{succ}} \mathcal{T} \tilde{\Pi}_{a,b} \Pi_{\leq k} |\phi\rangle\|$. This quantity is at most $\sum_{a+b \geq 2(K-k)} \left(\sqrt{\frac{2K}{N}} \right)^{a+b} \|\tilde{\Pi}_{a,b} \Pi_{\leq k} |\phi\rangle\|$ by Equation (10.3). Finally, by Cauchy-Schwarz' inequality and the fact that the supports of the projectors $\tilde{\Pi}_{a,b}$ are disjoint, we have $\|\Pi_{\text{succ}} \mathcal{T} \Pi_{\leq k} |\phi\rangle\| \leq \sqrt{\sum_{a+b \geq 2(K-k)} \left(\frac{2K}{N} \right)^{a+b}} \sqrt{\sum_{a,b} \|\tilde{\Pi}_{a,b} \Pi_{\leq k} |\phi\rangle\|^2} \leq N \left(\frac{2K}{N} \right)^{K-k} \|\Pi_{\leq k} |\phi\rangle\|$. \square

We can now conclude the proof of the main result of this section.

Theorem 10.4.6. *The success probability of finding K disjoint collisions in a random function $f : [M] \rightarrow [N]$ is at most $O(T^3/(K^2 N))^{K/2} + 2^{-K}$ for any algorithm making T quantum queries to f and any $1 \leq K \leq N/8$.*

Proof. Let $|\psi_T\rangle$ (resp. $|\phi_T\rangle$) denote the state of the algorithm after T queries in the standard (resp. recording) query model. We recall that $|\psi_T\rangle = \mathcal{T} |\phi_T\rangle$ (Theorem 10.3.3). Thus, by the triangle inequality, the success probability $\sigma = \|\Pi_{\text{succ}} |\psi_T\rangle\|^2$ satisfies $\sqrt{\sigma} \leq \|\Pi_{\text{succ}} \mathcal{T} \Pi_{\geq K/2} |\phi_T\rangle\| + \|\Pi_{\text{succ}} \mathcal{T} \Pi_{\leq K/2} |\phi_T\rangle\| \leq \|\Pi_{\geq K/2} |\phi_T\rangle\| + \|\Pi_{\text{succ}} \mathcal{T} \Pi_{\leq K/2} |\phi_T\rangle\|$. Using Propositions 10.4.4 and 10.4.5, we have that $\sqrt{\sigma} \leq \binom{T}{K/2} (4\sqrt{T/N})^{K/2} + N(2K/N)^{K/2} \leq O(T^{3/2}/(K\sqrt{N}))^{K/2} + 2^{-K/2-1}$. Finally, the upper bound on σ is derived from the standard inequality $(u+v)^2 \leq 2u^2 + 2v^2$. \square

10.5 Time lower bound for K -Search

In this section, we illustrate the use of the recording query model to upper bound the success probability of a query-bounded algorithm on a *non-uniform* input distribution.

Theorem 10.5.1. *The success probability of finding $K \leq N/8$ preimages of 1 in a random function $f : [M] \rightarrow \{0, 1\}$ where $f(x) = 1$ with probability K/N for each $x \in [M]$ is at most $O(T^2/(KN))^{K/2} + 2^{-K}$ for any algorithm using T quantum queries to f .*

We show that, similarly to the classical setting where a query can reveal a 1 with probability K/N , the *amplitude* of the basis states that record a new 1 increases by a factor of $\sqrt{K/N}$ after each query (Proposition 10.5.5). Thus, the amplitude of the basis states that have recorded at least $K/2$ ones after T queries is at most $O(T/\sqrt{KN})^{K/2}$. This implies that any algorithm with $T < O(\sqrt{KN})$ queries is likely to output at least $K/2$ ones at positions that have not been recorded. These outputs can only be correct with probability $O(K/N)^{K/2}$ (Proposition 10.5.6).

10.5.1 Recording query operator

We describe a recording operator that encodes the distribution that gives $f : [M] \rightarrow \{0, 1\}$ where $f(x) = 1$ with probability K/N independently for each $x \in [M]$. In the standard query model, the oracle's initial state is $|\text{init}\rangle = \otimes_{x \in [M]} (\sqrt{1 - K/N}|0\rangle_{\mathcal{F}_x} + \sqrt{K/N}|1\rangle_{\mathcal{F}_x})$ for this distribution. Consequently, we instantiate the recording query model as follows.

Definition 10.5.2. For any $x \in [M]$, define the unitary \mathcal{S}_x acting on the register \mathcal{F}_x to be

$$\mathcal{S}_x|\perp\rangle_{\mathcal{F}_x} = |+\rangle_{\mathcal{F}_x}, \quad \mathcal{S}_x|+\rangle_{\mathcal{F}_x} = |\perp\rangle_{\mathcal{F}_x}, \quad \mathcal{S}_x|-\rangle_{\mathcal{F}_x} = |-\rangle_{\mathcal{F}_x}$$

where $\alpha = \sqrt{1 - K/N}$, $\beta = \sqrt{K/N}$ and $|+\rangle_{\mathcal{F}_x} = \alpha|0\rangle_{\mathcal{F}_x} + \beta|1\rangle_{\mathcal{F}_x}$, $|-\rangle_{\mathcal{F}_x} = \beta|0\rangle_{\mathcal{F}_x} - \alpha|1\rangle_{\mathcal{F}_x}$.

We have $\mathcal{T}|\perp^M\rangle = |\text{init}\rangle$ when $\mathcal{T} = \otimes_{x \in [M]} \mathcal{S}_x$ as required by Theorem 10.3.3. The recording query operator is $\mathcal{R} = \mathcal{S}\mathcal{O}\mathcal{S}$ since $\mathcal{S}^\dagger = \mathcal{S}$, and it satisfies the next equations.

Lemma 10.5.3. *If the recording query operator \mathcal{R} associated with Definition 10.5.2 is applied to a basis state $|x, p, w\rangle|f\rangle$ where $p = 1$ then the register $|f(x)\rangle_{\mathcal{F}_x}$ is mapped to*

$$\begin{cases} (1 - 2\beta^2)|\perp\rangle + 2\alpha\beta^2|0\rangle - 2\alpha^2\beta|1\rangle & \text{if } f(x) = \perp \\ 2\alpha\beta^2|\perp\rangle + (1 - 2\alpha^2\beta^2)|0\rangle + 2\alpha^3\beta|1\rangle & \text{if } f(x) = 0 \\ -2\alpha^2\beta|\perp\rangle + 2\alpha^3\beta|0\rangle + (1 - 2\alpha^4)|1\rangle & \text{if } f(x) = 1 \end{cases}$$

and the other registers are unchanged. If $p = 0$ then none of the registers are changed.

Proof. By definition, the unitary \mathcal{S}_x maps $|\perp\rangle_{\mathcal{F}_x} \mapsto |+\rangle$, $|0\rangle_{\mathcal{F}_x} \mapsto \alpha|\perp\rangle + \beta|-\rangle$, $|1\rangle_{\mathcal{F}_x} \mapsto \beta|\perp\rangle - \alpha|-\rangle$. Thus, the action on the register \mathcal{F}_x is

- If $f(x) = \perp$ then $|f(x)\rangle_{\mathcal{F}_x} \xrightarrow{\mathcal{S}} |+\rangle \xrightarrow{\mathcal{O}} \alpha|0\rangle - \beta|1\rangle \xrightarrow{\mathcal{S}} (\alpha^2 - \beta^2)|\perp\rangle + 2\alpha\beta|-\rangle$.
- If $f(x) = 0$ then $|f(x)\rangle_{\mathcal{F}_x} \xrightarrow{\mathcal{S}} \alpha|\perp\rangle + \beta|-\rangle \xrightarrow{\mathcal{O}} \alpha|\perp\rangle + \beta(\beta|0\rangle + \alpha|1\rangle) \xrightarrow{\mathcal{S}} 2\alpha\beta^2|\perp\rangle + (1 - 2\alpha^2\beta^2)|0\rangle + 2\alpha^3\beta|1\rangle$.
- If $f(x) = 1$ then $|f(x)\rangle_{\mathcal{F}_x} \xrightarrow{\mathcal{S}} \beta|\perp\rangle - \alpha|-\rangle \xrightarrow{\mathcal{O}} \beta|\perp\rangle - \beta(\beta|0\rangle + \alpha|1\rangle) \xrightarrow{\mathcal{S}} -2\alpha^2\beta|\perp\rangle + 2\alpha^3\beta|0\rangle + (1 - 2\alpha^4)|1\rangle$.

□

If $\alpha \gg \beta$, the above lemma shows that \mathcal{R} is close to the mapping $|\perp\rangle_{\mathcal{F}_x} \mapsto |\perp\rangle - 2\beta|1\rangle$, $|0\rangle_{\mathcal{F}_x} \mapsto |0\rangle + 2\beta|1\rangle$, $|1\rangle_{\mathcal{F}_x} \mapsto -|1\rangle + 2\beta(|0\rangle - |\perp\rangle)$ up to lower order terms of amplitude $O(\beta^2)$.

10.5.2 Analysis of the recording progress

The measure of progress is based on the number of ones contained in the oracle's register. We first give some projectors related to this quantity.

Definition 10.5.4. We define the following projectors by giving the basis states on which they project:

- $\Pi_{\leq k}$, $\Pi_{=k}$ and $\Pi_{\geq k}$: all basis states $|x, p, w\rangle|f\rangle$ such that f contains respectively *at most*, *exactly* or *at least* k coordinates equal to 1.
- $\Pi_{=k, \perp}$ and $\Pi_{=k, 0}$: all basis states $|x, p, w\rangle|f\rangle$ such that (1) f contains *exactly* k coordinates equal to 1, (2) the phase multiplier is $p = 1$ and (3) $f(x) = \perp$ or $f(x) = 0$ respectively.

We can now define the measure of progress $q_{t,k}$ for t queries and k ones as

$$q_{t,k} = \|\Pi_{\geq k}|\phi_t\rangle\|$$

where $|\phi_t\rangle$ is the state after t queries in the recording query model. The main result of this section is the following bound on the growth of $q_{t,k}$.

Proposition 10.5.5. *For all t and k , we have that $q_{t,k} \leq \binom{t}{k} \left(\frac{4\sqrt{K}}{\sqrt{N}}\right)^k$.*

Proof. First, $q_{0,0} = 1$ and $q_{0,k} = 0$ for all $k \geq 1$ since the initial state is $|\phi_0\rangle = |\mathbf{0}\rangle|\perp^M\rangle$. Then, we prove that $q_{t,k}$ satisfies the following recurrence relation

$$q_{t+1,k+1} \leq q_{t,k+1} + 4\sqrt{\frac{K}{N}}q_{t,k}. \quad (10.4)$$

From this result, it is trivial to conclude that $q_{t,k} \leq \binom{t}{k} \left(\frac{4\sqrt{K}}{\sqrt{N}}\right)^k$. In order to prove Equation (10.4), we first observe that $q_{t+1,k+1} = \|\Pi_{\geq k+1}U_{t+1}\mathcal{R}|\phi_t\rangle\| = \|\Pi_{\geq k+1}\mathcal{R}|\phi_t\rangle\|$ where U_{t+1} is the unitary applied by the algorithm at time $t+1$. Then, on a basis state $|x, p, w\rangle|f\rangle$, the recording query operator \mathcal{R} acts as the identity on the registers $\mathcal{F}_{x'}$ for $x' \neq x$. Consequently, the basis states $|x, p, w\rangle|f\rangle$ in $|\phi_t\rangle$ that may contribute to $q_{t+1,k+1}$ must either already contain $k+1$ ones in f , or exactly k ones in f and $f(x) \neq 1, p = 1$. This implies that

$$q_{t+1,k+1} \leq q_{t,k+1} + \|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,\perp}|\phi_t\rangle\| + \|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,0}|\phi_t\rangle\|.$$

We first bound the term $\|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,\perp}|\phi_t\rangle\|$. Consider any state $|x, p, w\rangle|f\rangle$ in the support of $\Pi_{=k,\perp}$. By Lemma 10.5.3, we have $\Pi_{\geq k+1}\mathcal{R}|x, p, w\rangle|f\rangle = -2\alpha^2\beta|x, p, w\rangle|1\rangle_{\mathcal{F}_x} \otimes_{x' \neq x} |f(x')\rangle_{\mathcal{F}_{x'}}$. Since any two basis states in the support of $\Pi_{=k,\perp}$ remain orthogonal after $\Pi_{\geq k+1}\mathcal{R}$ is applied, we obtain that $\|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,\perp}|\phi_t\rangle\| = 2\alpha^2\beta\|\Pi_{=k,\perp}|\phi_t\rangle\| \leq 2\sqrt{K/N}(1 - K/N)q_{t,k}$.

Similarly, for $|x, p, w\rangle|f\rangle$ in the support of $\Pi_{=k,0}$ we have $\|\Pi_{\geq k+1}\mathcal{R}|x, p, w\rangle|f\rangle\| = 2\alpha^3\beta$ by Lemma 10.5.3. Consequently, $\|\Pi_{\geq k+1}\mathcal{R}\Pi_{=k,0}|\phi_t\rangle\| = 2\alpha^3\beta\|\Pi_{=k,0}|\phi_t\rangle\| \leq 2\sqrt{K/N}(1 - K/N)^{3/2}q_{t,k}$. We can now conclude the proof,

$$q_{t+1,k+1} \leq q_{t,k+1} + 2\sqrt{\frac{K}{N}}\left(1 - \frac{K}{N}\right)q_{t,k} + 2\sqrt{\frac{K}{N}}\left(1 - \frac{K}{N}\right)^{3/2}q_{t,k} \leq q_{t,k+1} + 4\sqrt{\frac{K}{N}}q_{t,k}.$$

□

10.5.3 From the recording progress to the success probability

We connect the success probability $\sigma = \|\Pi_{\text{succ}}|\psi_T\rangle\|^2$ in the standard query model to the final progress $q_{T,k}$ in the recording query model after T queries. We show that if the algorithm has made no significant progress for $k \geq K/2$ then it needs to “guess” that $f(x) = 1$ for about $K - k$ positions where the \mathcal{F}_x register does not contain 1. Classically, the probability to find $K - k$ preimages of 1 at positions that have not been queried would be $(K/N)^{K-k}$. Here, we show similarly that if a unit state contains at most k ones in the quantum recording model, then after mapping it to the standard query model (by applying the operator \mathcal{T} of Theorem 10.3.3) the probability that the output register contains the correct positions of K preimages of 1 is at most $3^K \left(\frac{K}{N}\right)^{K-k}$.

Proposition 10.5.6. *For any $|\phi\rangle$, we have $\|\Pi_{\text{succ}}\mathcal{T}\Pi_{\leq k}|\phi\rangle\| \leq 3^{K/2} \left(\sqrt{\frac{K}{N}}\right)^{K-k} \|\Pi_{\leq k}|\phi\rangle\|$.*

Proof. Let $|x, p, w\rangle|f\rangle$ be any basis state in the support of $\Pi_{\leq k}$. The output value z is a substring of w made of K distinct values $x_1, \dots, x_K \in [M]$ indicating positions where the input f is supposed to contain ones. By definition of $\Pi_{\leq k}$, we have $f(x_i) \neq 1$ for at least $K - k$ indices $i \in [K]$. For each such index i , after applying $\mathcal{T} = \otimes_{x' \in [M]} \mathcal{S}_{x'}$, the amplitude of $|1\rangle_{\mathcal{F}_{x_i}}$ is $\sqrt{\frac{K}{N}}$ (if $f(x_i) = \perp$) or $\sqrt{\frac{K}{N}(1 - \frac{K}{N})}$ (if $f(x_i) = 0$) by Definition 10.5.2. Consequently,

$$\|\Pi_{\text{succ}}\mathcal{T}|x, p, w\rangle|f\rangle\| \leq \left(\sqrt{\frac{K}{N}}\right)^{K-k}. \quad (10.5)$$

Fix any state $|\phi\rangle$ and denote $|\varphi\rangle = \Pi_{\leq k}|\phi\rangle = \sum_{x,p,w,f} \alpha_{x,p,w,f} |x, p, w\rangle|f\rangle$. Let us write $w_{\vec{x}} = \{x_1, \dots, x_K\}$ when the output substring z of w contains x_1, \dots, x_K . For any two basis states $|x, p, w\rangle|f\rangle$ and $|\bar{x}, \bar{p}, \bar{w}\rangle|\bar{f}\rangle$, if $(x, p, w, (f(x'))_{x' \notin w_{\vec{x}}}) \neq (\bar{x}, \bar{p}, \bar{w}, (\bar{f}(x'))_{x' \notin w_{\vec{x}}})$ then $\Pi_{\text{succ}}\mathcal{T}|x, p, w\rangle|f\rangle$ is orthogonal to $\Pi_{\text{succ}}\mathcal{T}|\bar{x}, \bar{p}, \bar{w}\rangle|\bar{f}\rangle$. There are 3^K choices for $|x, p, w\rangle|f\rangle$ once we set the value of $(x, p, w, (f(x'))_{x' \notin w_{\vec{x}}})$ since it remains to choose $f(x') \in \{\perp, 0, 1\}$ for $x' \in w_{\vec{x}}$. By using the Cauchy–Schwarz inequality and Equation (10.5), we get that

$$\begin{aligned} \|\Pi_{\text{succ}}\mathcal{T}|\varphi\rangle\|^2 &= \sum_{x,p,w,(f(x'))_{x' \notin w_{\vec{x}}}} \left\| \sum_{(f(x'))_{x' \in w_{\vec{x}}}} \alpha_{x,p,w,f} \Pi_{\text{succ}}\mathcal{T}|x, p, w\rangle|f\rangle \right\|^2 \\ &\leq \sum_{x,p,w,(f(x'))_{x' \notin w_{\vec{x}}}} \left(\sum_{(f(x'))_{x' \in w_{\vec{x}}}} |\alpha_{x,p,w,f}|^2 \right) \left(\sum_{(f(x'))_{x' \in w_{\vec{x}}}} \|\Pi_{\text{succ}}\mathcal{T}|x, p, w\rangle|f\rangle\|^2 \right) \\ &\leq \|\varphi\|^2 \cdot 3^K \left(\frac{K}{N}\right)^{K-k}. \end{aligned}$$

□

We can now conclude the proof of the main result.

Proof of Theorem 10.5.1. Let $|\psi_T\rangle$ (resp. $|\phi_T\rangle$) denote the state of the algorithm after T queries in the standard (resp. recording) query model. According to Theorem 10.3.3, we have $|\psi_T\rangle = \mathcal{T}|\phi_T\rangle$. Thus, by the triangle inequality, the success probability $\sigma = \|\Pi_{\text{succ}}|\psi_T\rangle\|^2$ satisfies $\sqrt{\sigma} \leq \|\Pi_{\text{succ}}\mathcal{T}\Pi_{\geq K/2}|\phi_T\rangle\| + \|\Pi_{\text{succ}}\mathcal{T}\Pi_{\leq K/2}|\phi_T\rangle\| \leq \|\Pi_{\geq K/2}|\phi_T\rangle\| + \|\Pi_{\text{succ}}\mathcal{T}\Pi_{\leq K/2}|\phi_T\rangle\|$. Using Propositions 10.5.5 and 10.5.6, we have that $\sqrt{\sigma} \leq \binom{T}{K/2} (4\sqrt{K/N})^{K/2} + 3^{K/2} (\sqrt{K/N})^{K/2} \leq O(T/\sqrt{KN})^{K/2} + 2^{-K/2-1}$. Finally, the upper bound on σ is derived from the standard inequality $(u + v)^2 \leq 2u^2 + 2v^2$. □

10.6 Time-space tradeoffs

10.6.1 Time-space tradeoff for Collision Pairs Finding

We use the time lower bound obtained in Section 10.4 to derive a new time-space tradeoff for the problem of finding K disjoint collisions in a random function $f : [M] \rightarrow [N]$. We recall that the output is produced in an online fashion (Section 10.2.2), meaning that a collision can be output as soon as it is discovered. The length of the output is not counted toward the space bound. We allow the same collision to be output several times, but it contributes only once to the total count.

Theorem 10.6.1. *Any quantum algorithm for finding K disjoint collisions in a random function $f : [M] \rightarrow [N]$ with success probability $2/3$ must satisfy a time-space tradeoff of $T^3 S \geq \Omega(K^3 N)$, where $1 \leq K \leq N/8$.*

Proof. Our proof relies on the time-segmentation method for large-output problems used in [BFK+81; KŠW07] for instance. Fix any quantum circuit \mathcal{C} in the space-bounded model of Section 10.2.2 running in time T and using $S > \Omega(\log N)$ qubits of memory. The circuit \mathcal{C} is partitioned into $L = T/T'$ consecutive sub-circuits $\mathcal{C}_1 \parallel \mathcal{C}_2 \parallel \dots \parallel \mathcal{C}_L$ each running in time $T' = S^{2/3} N^{1/3}$, where \mathcal{C}_j takes as input the output memory of \mathcal{C}_{j-1} for each $j \in [L]$. Define X_j to be the random variable that counts the number of (mutually) disjoint collisions that \mathcal{C} outputs between time $(j-1)T'$ and jT' (i.e. in the sub-circuit \mathcal{C}_j) when the input is a random function $f : [M] \rightarrow [N]$. The algorithm must satisfy $\sum_{j=1}^L \mathbb{E}[X_j] \geq \Omega(K)$ to be correct. We claim that the algorithm outputs at most $3S$ collisions in expectation in each segment of the computation. Assume by contradiction that $\mathbb{E}[X_j] \geq 3S$ for some j . Since X_j is bounded between 0 and N we have $\Pr[X_j > 2S] \geq S/N$. Consequently, by running \mathcal{C}_j on the completely mixed state on S qubits we obtain $2S$ disjoint collisions with probability at least $S/N \cdot 2^{-S}$ in time T' (this is akin to a union bound argument). However, by Theorem 10.4.6, no quantum algorithm can find more than $2S$ disjoint collisions in time $T' = S^{2/3} N^{1/3}$ with success probability larger than 4^{-S+1} . This contradiction implies that $\mathbb{E}[X_j] \leq 3S$ for all j . Consequently, there must be at least $L \geq \Omega(K/S)$ sub-circuits in order to have $\sum_{j=1}^L \mathbb{E}[X_j] \geq \Omega(K)$. Since each sub-circuit runs in time $S^{2/3} N^{1/3}$ the running time of \mathcal{C} is $T \geq \Omega(L \cdot S^{2/3} N^{1/3}) \geq \Omega(K N^{1/3} / S^{1/3})$. \square

As an illustration of the above result, we obtain that any quantum algorithm for finding $N/8$ disjoint collisions in a random function must satisfy a time-space tradeoff of $T S^{1/3} \geq \Omega(N^{4/3})$. We prove that any improvement to this lower bound would imply a breakthrough for the Element Distinctness problem.

Definition 10.6.2. The Element Distinctness problem ED_N on domain size N consists of finding a collision in a random function $f : [N] \rightarrow [N^2]$.

It is well-known that the query complexity of Element Distinctness is $T = \Theta(N^{2/3})$ [AS04; Amb07]. However, it is a long-standing open problem to find any quantum time-space lower bound (even classically the question is not completely settled yet [Yao94; BSSV03]). Here, we show that *any* improvement to Theorem 10.6.1 would imply a non-trivial time-space tradeoff for Element Distinctness. This result relies on a reduction presented in Algorithm 10.1 and analyzed in Proposition 10.6.3 (the constants c_0, c_1, c_2 are chosen in the proof).

Proposition 10.6.3. *Let N be a square number. If there is an algorithm solving ED_N in time T_N and space S_N then Algorithm 10.1 runs in time $O(NT_{\sqrt{N}})$ and space $O(S_{\sqrt{N}})$, and it finds $c_1 N$ collisions in any function $f : [N] \rightarrow [N]$ containing at least $c_0 N$ collisions.*

Input: a function $f : [N] \rightarrow [N]$ containing at least $c_0 N$ collisions.

Output: at least $c_1 N$ collisions in f (not necessarily disjoint).

1. Repeat $c_2 N$ times:

- a) Sample a 4-wise independent hash function $h : [\sqrt{N}] \rightarrow [N]$ and store it in memory.
- b) Run an algorithm for $\text{ED}_{\sqrt{N}}$ on input $f \circ h : [\sqrt{N}] \rightarrow [N]$. If it finds a collision $(f \circ h(i), f \circ h(j))$ check if $h(i) \neq h(j)$ and output the collision $(h(i), h(j))$ in this case.

Algorithm 10.1: Finding collisions by using $\text{ED}_{\sqrt{N}}$.

Proof. We choose $c_0 = 40$, $c_1 = 1/10^4$ and $c_2 = 8$. We study the probabilities of the following events to occur in a fixed round of Algorithm 10.1:

- **Event A:** The hash function h is collision free (i.e. $h(i) \neq h(j)$ for all $i \neq j$).
- **Event B:** None of the collisions output during the previous rounds is present in the image of h .
- **Event C:** The function $f \circ h : [\sqrt{N}] \rightarrow [N]$ contains a collision.
- **Event D:** The algorithm for $\text{ED}_{\sqrt{N}}$ finds a collision at step 2.b.

Algorithm 10.1 succeeds if and only if the event $A \wedge B \wedge C \wedge D$ occurs during at least $c_1 N$ rounds. We now lower bound the probability of this event happening.

For **event A**, let us consider the random variable $X = \sum_{i \neq j \in [\sqrt{N}]} 1_{h(i)=h(j)}$. Using that h is pairwise independent, we have $\mathbb{E}[X] = \binom{\sqrt{N}}{2} \frac{1}{N} \leq \frac{1}{2}$. Thus, by Markov's inequality, $\Pr[A] = 1 - \Pr[X \geq 1] \geq \frac{1}{2}$.

For **event B**, let us assume that $k < c_1 N$ collisions $(x_1, x_2), \dots, (x_{2k-1}, x_{2k})$ have been output so far. For any $i \in [k]$, the probability that both x_{2i-1} and x_{2i} belong to $\{h(1), \dots, h(\sqrt{N})\}$ is at most $\binom{\sqrt{N}}{2} \frac{2}{N^2} \leq \frac{1}{N}$ since h is pairwise independent. By a union bound, $\Pr[B] \geq 1 - \frac{k}{N} \geq 1 - c_1$.

For **event C**, let us consider the binary random variables $Y_{i,j} = 1_{f \circ h(i)=f \circ h(j)}$ for $i \neq j \in [\sqrt{N}]$, and let $Y = \sum_{i \neq j} Y_{i,j}$ be twice the number of collisions in $f \circ h$. Note that we may have $Y_{i,j} = 1$ because $h(i) = h(j)$ (this is taken care of in event A). For each $y \in [N]$, let $N_y = |\{x : f(x) = y\}|$ denote the number of elements that are mapped to y by f . Using that h is 4-wise independent, for any $i \neq j \neq k \neq \ell$ we have,

$$\begin{cases} \Pr[Y_{i,j} = 1] = \frac{\sum_{y \in [N]} N_y^2}{N^2} \\ \Pr[Y_{i,j} = 1 \wedge Y_{i,k} = 1] = \frac{\sum_{y \in [N]} N_y^3}{N^3} \\ \Pr[Y_{i,j} = 1 \wedge Y_{k,\ell} = 1] = \Pr[Y_{i,j} = 1] \cdot \Pr[Y_{k,\ell} = 1]. \end{cases}$$

Consequently, $\mathbb{E}[Y] = \binom{\sqrt{N}}{2} \frac{\sum_{y \in [N]} N_y^2}{N^2}$ and

$$\begin{aligned} \text{Var}[Y] &= \sum_{\{i,j\}} \text{Var}[Y_{i,j}] + \sum_{\{i,j\} \neq \{i,k\}} \text{Cov}[Y_{i,j}, Y_{i,k}] + \sum_{\{i,j\} \cap \{k,\ell\} = \emptyset} \text{Cov}[Y_{i,j}, Y_{k,\ell}] \\ &\leq \sum_{\{i,j\}} \mathbb{E}[Y_{i,j}^2] + \sum_{\{i,j\} \neq \{i,k\}} \mathbb{E}[Y_{i,j} Y_{i,k}] \\ &= \binom{\sqrt{N}}{2} \frac{\sum_{y \in [N]} N_y^2}{N^2} + 3 \binom{\sqrt{N}}{3} \frac{\sum_{y \in [N]} N_y^3}{N^3} \end{aligned}$$

where we have used that $Y_{i,j}$ and $Y_{k,\ell}$ are independent when $i \neq j \neq k \neq \ell$. The term $\sum_{y \in [N]} N_y^2$ is equal to the number of pairs $(x, x') \in [N]^2$ such that $f(x) = f(x')$. Each collision in f gives two such pairs, and we must also count the pairs (x, x) . Thus, $\sum_{y \in [N]} N_y^2 \geq (1 + 2c_0)N$. Moreover, $\sum_{y \in [N]} N_y^3 \leq (\sum_{y \in [N]} N_y^2)^{3/2}$. Consequently,

$$\frac{\text{Var}[Y]}{\mathbb{E}[Y]^2} \leq \frac{1 + \sqrt{N} \left(\frac{\sum_{y \in [N]} N_y^2}{N^2} \right)^{1/2}}{\left(\binom{\sqrt{N}}{2} \frac{\sum_{y \in [N]} N_y^2}{N^2} \right)^2} \leq \frac{4(1 + \sqrt{1 + 2c_0})}{1 + 2c_0}.$$

Finally, according to Chebyshev's inequality, $\Pr[Y = 0] \leq \Pr[|Y - \mathbb{E}[Y]| \geq \mathbb{E}[Y]] \leq \frac{\text{Var}[Y]}{\mathbb{E}[Y]^2}$.

Thus, $\Pr[C] = 1 - \Pr[Y = 0] \geq 1 - \frac{4(1 + \sqrt{1 + 2c_0})}{1 + 2c_0}$.

For **event D**, we have $\Pr[D \mid A \wedge B \wedge C] \geq 2/3$ assuming the bounded-error algorithm for solving $\text{ED}_{\sqrt{N}}$ succeeds with probability $2/3$.

The probability of the four events happening together is $\Pr[A \wedge B \wedge C \wedge D] = \Pr[D \mid A \wedge B \wedge C] \cdot \Pr[A \wedge B \wedge C] \geq \Pr[D \mid A \wedge B \wedge C] \cdot (\Pr[A] + \Pr[B] + \Pr[C] - 2) \geq \frac{2}{3} \cdot \left(\frac{1}{2} - c_1 - \frac{4(1 + \sqrt{1 + 2c_0})}{1 + 2c_0} \right) \geq 1/250$. Let τ be the number of rounds after which $c_1 N$ collisions have been found (i.e. $A \wedge B \wedge C \wedge D$ has occurred $c_1 N$ times). We have $\mathbb{E}[\tau] \leq 8c_1 N$, and by Markov's inequality $\Pr[\tau \geq c_2 N] \leq 250c_1/c_2 \leq 1/3$. Thus, with probability at least $2/3$, Algorithm 10.1 outputs at least $c_1 N$ collisions in f . \square

We now use the above reduction to transform any low-space algorithm for Element Distinctness into one for finding $\Omega(N/\log N)$ disjoint collisions in a random function. Observe that Algorithm 10.1 does not necessarily output collisions that are mutually disjoint. Nevertheless, there is a small probability that a random function $f : [M] \rightarrow [N]$ contains multi-collisions of size larger than $\log N$ when $M \approx N$ [FO89]. Thus, there is only a $\log N$ loss in the analysis.

Proposition 10.6.4. *Suppose that there exists a bounded-error quantum algorithm for solving Element Distinctness on domain size N that satisfies a time-space tradeoff of $T^\alpha S^\beta \leq \tilde{O}(N^{2(\gamma-\alpha)})$ for some constants α, β, γ . Then, there exists a bounded-error quantum algorithm for finding $\Omega(N/\log N)$ disjoint collisions in a random function $f : [10N] \rightarrow [N]$ that satisfies a time-space tradeoff of $T^\alpha S^\beta \leq \tilde{O}(N^\gamma)$.*

Proof. We use the constants c_0, c_1, c_2 specified in the proof of Proposition 10.6.3. First, we note that a random function $f : [10N] \rightarrow [N]$ contains $c_0 N$ collisions and no multi-collisions of size larger than $\log(N)$ with large probability [FO89]. Consequently, any set of $c_1 N$ collisions must contain at least $c_1 N / \log N$ mutually disjoint collisions with large probability. Assume now that there exists an algorithm solving $\text{ED}_{\sqrt{10N}}$ in time $T_{\sqrt{10N}}$ and space $S_{\sqrt{10N}}$ such that $(T_{\sqrt{10N}})^\alpha S_{\sqrt{10N}}^\beta \leq \tilde{\Omega}(N^{\gamma-\alpha})$. Then, by plugging it

into Algorithm 10.1, one can find $c_1 N / \log N$ disjoint collisions in a random function $f : [10N] \rightarrow [N]$ in time $T = O(NT_{\sqrt{10N}})$ and space $S = O(S_{\sqrt{10N}})$. We derive from the above tradeoff that $T^\alpha S^\beta \leq \tilde{O}(N^\gamma)$. \square

As an application of Proposition 10.6.4, we obtain the following result regarding the hardness of finding $\tilde{\Omega}(N)$ collisions.

Corollary 10.6.5. *Suppose that there exists $\epsilon \in (0, 1)$ such that any quantum algorithm for finding $\tilde{\Omega}(N)$ disjoint collisions in a random function $f : [10N] \rightarrow [N]$ must satisfy a time-space tradeoff of $TS^{1/3} \geq \tilde{\Omega}(N^{4/3+\epsilon})$. Then, any quantum algorithm for solving Element Distinctness on domain size N must satisfy a time-space tradeoff of $TS^{1/3} \geq \tilde{\Omega}(N^{2/3+2\epsilon})$.*

We conjecture that the optimal tradeoff for finding K collisions is $T^2 S = \Theta(K^2 N)$, which would imply an optimal time-space tradeoff of $T^2 S \geq \tilde{\Omega}(N^2)$ for Element Distinctness.

Conjecture 1. Any quantum algorithm for finding K disjoint collisions in a random function $f : [M] \rightarrow [N]$ with success probability $2/3$ must satisfy a time-space tradeoff of $T^2 S \geq \Omega(K^2 N)$.

Corollary 10.6.6. *If Conjecture 1 is true, then any quantum algorithm for solving the Element Distinctness problem with success probability $2/3$ must satisfy a time-space tradeoff of $T^2 S \geq \tilde{\Omega}(N^2)$.*

We describe a quantum algorithm that achieves the tradeoff of $T^2 S \leq \tilde{O}(K^2 N)$. In order to simplify the analysis, we do not require the collisions to be disjoint.

1. Repeat $\tilde{O}(K/S)$ times:
 - a) Sample a subset $G \subset [N]$ of size S uniformly at random.
 - b) Construct a table containing all pairs $(x, f(x))$ for $x \in G$. Sort the table according to the second entry of each pair.
 - c) Define the function $g : [N] \setminus G \rightarrow \{0, 1\}$ where $g(x) = 1$ iff there exists $x' \in G$ such that $f(x) = f(x')$. Run the Grover search algorithm [BBHT98] on g , by using the table computed at step 1.b, to find all pairs $(x, x') \in G \times ([N] \setminus G)$ such that $f(x) = f(x')$. Output all of these pairs.

Algorithm 10.2: Finding K collision pairs in $f : [N] \rightarrow [N]$ using a memory of size S .

Proposition 10.6.7. *For any $1 \leq K \leq O(N)$ and $\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K^{2/3} N^{1/3})$, there exists a bounded-error quantum algorithm that can find K collisions in a random function $f : [N] \rightarrow [N]$ by making $T = \tilde{O}(K\sqrt{N/S})$ queries and using S qubits of memory.*

Proof. We prove that Algorithm 10.2 satisfies the statement of the proposition. For simplicity, we do not try to tune the hidden factors in the big O notations.

The probability that a fixed pair (x, x') satisfies $(x, x') \in G \times ([N] \setminus G)$ for at least one iteration of step 1 is $\Omega(K/S \cdot S/N \cdot (1 - S/N)) = \Omega(K/N)$. Since a random function $f : [N] \rightarrow [N]$ contains $\Omega(N)$ collisions with high probability, the algorithm encounters $\Omega(K)$ collisions in total. Thus, if the Grover search algorithm never fails we obtain the desired number of collisions.

The expected number of pre-images of 1 under g is $O(S)$. Consequently, the complexity of Grover's search at step 1.c is $O(\sqrt{SN})$. The overall query complexity is $T = \tilde{O}(K/S \cdot \sqrt{SN}) = \tilde{O}(K\sqrt{N/S})$, and the space complexity is $\tilde{O}(S)$. \square

10.6.2 Time-space tradeoff for Sorting

We use the time lower bound obtained in Section 10.5 to reprove the time-space tradeoff for the Sorting problem described in [KŠW07, Theorem 21]. The input to the Sorting problem is represented as a function $f : [N] \rightarrow \{0, 1, 2\}$ (we do not need to consider a larger range for the proof). A quantum algorithm for the Sorting problem must output in order a sequence $x_1, \dots, x_N \in [N]$ of distinct integers such that $f(x_1) \geq f(x_2) \geq \dots \geq f(x_N)$ with probability at least $2/3$.

Theorem 10.6.8. *Any quantum algorithm for sorting a function $f : [N] \rightarrow \{0, 1, 2\}$ with success probability $2/3$ must satisfy a time-space tradeoff of $T^2 S \geq \Omega(N^3)$.*

Proof. The proof is a modified version of [KŠW07, Theorem 21] adapted to our version of the K -Search problem. Given a circuit \mathcal{C} that runs in time T and space $\Omega(\log N) \leq S \leq N/64$, we partition it into $L = T/T'$ consecutive sub-circuits $\mathcal{C}_1 \parallel \mathcal{C}_2 \parallel \dots \parallel \mathcal{C}_L$ each running in time $T' = \sqrt{SN}/4$. Assume by contradiction that a circuit \mathcal{C}_j outputs the elements of ranks $r, r+1, \dots, r+2S-1$ for some $r \leq N/2$. We use \mathcal{C}_j to solve the K -search problem for $K = 2S$ as follows. Given an input $g : [N/2] \rightarrow \{0, 1\}$ to the K -search problem where $g(x) = 1$ with probability $\frac{K}{N/4}$ for each x , define the function $f : [N] \rightarrow \{0, 1, 2\}$ where

$$f(x) = \begin{cases} 2 & \text{if } x < r, \\ g(x - r + 1) & \text{if } r \leq x < r + N/2, \\ 0 & \text{if } x \geq r + N/2. \end{cases}$$

Note that the function g contains at least $2S$ preimages of 1 with probability at least $2S/N$. Thus, if the circuit \mathcal{C} is run on the input f , then the indices output by the sub-circuit \mathcal{C}_j must contain the position of $2S$ preimages of 1 with probability at least $2/3 \cdot 2S/N$. Consequently, by running \mathcal{C}_j on the completely mixed state on S qubits we can find $2S$ preimages of 1 under g with probability at least $2/3 \cdot 2S/N \cdot 2^{-S}$ in time T' . However, by Theorem 10.5.1, any such algorithm must succeed with probability at most 4^{-S+1} . This contradiction implies that there must be at least $L \geq \Omega(N/S)$ sub-circuits in \mathcal{C} . Thus, the running time of \mathcal{C} is $T \geq \Omega(L \cdot \sqrt{SN}) \geq \Omega(N^{3/2}/\sqrt{S})$. \square

The time-space tradeoffs for the Boolean matrix-vector product [KŠW07, Theorem 23] and the Boolean matrix product [KŠW07, Theorem 25] problems can be reproved in a similar way.

10.7 Discussion

We investigated the use of the quantum recording technique for proving time-space tradeoff lower bounds. As in previous work [KŠW07; AŠW09], we reduced this question to query complexity lower bounds in the exponentially small success probability regime by using a quantum union-bound argument. This method cannot give a better tradeoff than $T^3 S \geq \Omega(K^3 N)$ for finding K collision pairs. Indeed, the slicing technique used in Theorem 10.6.1 also applies to the quantum algorithms that must “compress” their memory to S qubits every $S^{2/3} N^{1/3}$ steps of computation, but that can use unlimited memory inside each slice. There is a simple algorithm in this setting that can find K collision pairs by using $T = \tilde{O}(KN^{1/3}/S^{1/3})$ quantum queries, which matches our lower bound. This is in stark contrast with the Sorting problem where having temporary unlimited memory does not help the computation. In order to improve the above tradeoff, it may be easier to first consider the comparison-based query model, where a near-optimal classical time-space tradeoff is known for the Element Distinctness problem [BFM+87; Yao94].

Bibliography

- [AA05] S. Aaronson and A. Ambainis. “Quantum Search of Spatial Regions”. In: *Theory of Computing* 1.4 (2005), pp. 47–79 (cit. on p. 19).
- [Aar05] S. Aaronson. “Limitations of Quantum Advice and One-Way Communication”. In: *Theory of Computing* 1.1 (2005), pp. 1–28 (cit. on p. 129).
- [Aar15] S. Aaronson. “Read the Fine Print”. In: *Nature Physics* 11.4 (2015), pp. 291–293 (cit. on pp. 87, 97).
- [ABC+20] S. Arunachalam, A. Belovs, A. M. Childs, R. Kothari, A. Rosmanis, and R. de Wolf. “Quantum Coupon Collector”. In: *Proceedings of the 15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*. 2020, 10:1–10:17 (cit. on p. 31).
- [Abr90] K. Abrahamson. “A Time-Space Tradeoff for Boolean Matrix Multiplication”. In: *Proceedings of the 31st Symposium on Foundations of Computer Science (FOCS)*. 1990, pp. 412–419 (cit. on p. 126).
- [AG19a] J. van Apeldoorn and A. Gilyén. “Improvements in Quantum SDP-Solving with Applications”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 99:1–99:15 (cit. on pp. 97, 98).
- [AG19b] J. van Apeldoorn and A. Gilyén. *Quantum Algorithms for Zero-Sum Games*. [arXiv:1904.03180](#) [quant-ph]. 2019 (cit. on p. 98).
- [AGGW20a] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf. “Convex Optimization Using Quantum Oracles”. In: *Quantum* 4 (2020), p. 220 (cit. on p. 98).
- [AGGW20b] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf. “Quantum SDP-Solvers: Better Upper and Lower Bounds”. In: *Quantum* 4 (2020), p. 230 (cit. on pp. 28, 31, 32, 98).
- [AHLW16] Y. Ai, W. Hu, Y. Li, and D. P. Woodruff. “New Characterizations in Turnstile Streams with Applications”. In: *Proceedings of the 31st Conference on Computational Complexity (CCC)*. 2016 (cit. on pp. 115, 121).
- [AHN+20] S. Arunachalam, V. Havlicek, G. Nannicini, K. Temme, and P. Wocjan. *Simpler (Classical) and Faster (Quantum) Algorithms for Gibbs Partition Functions*. [arXiv:2009.11270](#) [quant-ph]. 2020 (cit. on p. 43).
- [AK16] S. Arora and S. Kale. “A Combinatorial, Primal-Dual Approach to Semidefinite Programs”. In: *Journal of the ACM* 63.2 (2016), pp. 1–35 (cit. on p. 98).
- [AKN98] D. Aharonov, A. Kitaev, and N. Nisan. “Quantum Circuits with Mixed States”. In: *Proceedings of the 30th Symposium on Theory of Computing (STOC)*. 1998, pp. 20–30 (cit. on pp. 7, 17).

Bibliography

- [AKO10] A. Andoni, R. Krauthgamer, and K. Onak. *Streaming Algorithms from Precision Sampling*. [arXiv:1011.1263](#) [cs.DS]. 2010 (cit. on pp. 7, 116–118, 122, 123).
- [AL20] S. Apers and T. Lee. *Quantum Complexity of Minimum Cut*. [arXiv:2011.09823](#) [quant-ph]. 2020 (cit. on p. 98).
- [ALS20] B. Axelrod, Y. P. Liu, and A. Sidford. “Near-optimal Approximate Discrete and Continuous Submodular Function Minimization”. In: *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*. 2020, pp. 837–853 (cit. on pp. 98, 102, 103, 112).
- [AM20] S. Arunachalam and R. Maity. “Quantum Boosting”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020, pp. 377–387 (cit. on p. 98).
- [Amb02] A. Ambainis. “Quantum Lower Bounds by Quantum Arguments”. In: *Journal of Computer and System Sciences* 64.4 (2002), pp. 750–767 (cit. on pp. 126, 130).
- [Amb07] A. Ambainis. “Quantum Walk Algorithm for Element Distinctness”. In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239 (cit. on pp. 7, 125, 128, 140).
- [Amb10a] A. Ambainis. “A New Quantum Lower Bound Method, with an Application to a Strong Direct Product Theorem for Quantum Search”. In: *Theory of Computing* 6.1 (2010), pp. 1–25 (cit. on p. 126).
- [Amb10b] A. Ambainis. “Quantum Search with Variable Times”. In: *Theory of Computing Systems* 47.3 (2010), pp. 786–807 (cit. on p. 46).
- [Amb10c] A. Ambainis. *Variable Time Amplitude Amplification and a Faster Quantum Algorithm for Solving Systems of Linear Equations*. [arXiv:1010.4458](#) [quant-ph]. 2010 (cit. on pp. 46, 47, 49–51, 54).
- [Amb12] A. Ambainis. “Variable Time Amplitude Amplification and Quantum Algorithms for Linear Algebra Problems”. In: *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science (STACS)*. 2012, pp. 636–647 (cit. on pp. 4, 45–47).
- [AMRR11] A. Ambainis, L. Magnin, M. Roetteler, and J. Roland. “Symmetry-Assisted Adversaries for Quantum State Generation”. In: *Proceedings of the 26th Computational Complexity Conference (CCC)*. 2011 (cit. on p. 126).
- [AMRS20] G. Alagic, C. Majenz, A. Russell, and F. Song. “Quantum-Access-Secure Message Authentication via Blind-Unforgeability”. In: *Proceedings of the 39th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 2020, pp. 788–817 (cit. on p. 126).
- [AMS99] N. Alon, Y. Matias, and M. Szegedy. “The Space Complexity of Approximating the Frequency Moments”. In: *Journal of Computer and System Sciences* 58.1 (1999), pp. 137–147 (cit. on pp. 3, 7, 25, 116, 117, 123, 124).
- [And86] H. L. Anderson. “Metropolis, Monte Carlo, and the MANIAC”. In: *Los Alamos Science* 14 (1986), pp. 96–107 (cit. on p. 1).
- [Ape19] S. Apers. “Quantum Walk Sampling by Growing Seed Sets”. In: *Proceedings of the 27th European Symposium on Algorithms (ESA)*. 2019, 9:1–9:12 (cit. on p. 88).

- [AS04] S. Aaronson and Y. Shi. “Quantum Lower Bounds for the Collision and the Element Distinctness Problems”. In: *Journal of the ACM* 51.4 (2004), pp. 595–605 (cit. on pp. 126, 128, 140).
- [AŠW09] A. Ambainis, R. Špalek, and R. de Wolf. “A New Quantum Lower Bound Method, with Applications to Direct Product Theorems and Time-Space Tradeoffs”. In: *Algorithmica* 55.3 (2009), pp. 422–461 (cit. on pp. 7, 126, 131, 144).
- [AT07] D. Aharonov and A. Ta-Shma. “Adiabatic Quantum State Generation”. In: *SIAM Journal on Computing* 37.1 (2007), pp. 47–82 (cit. on pp. 5, 30, 88).
- [AW18] S. Arunachalam and R. de Wolf. “Optimal Quantum Sample Complexity of Learning Algorithms”. In: *Journal of Machine Learning Research* 19.1 (2018), pp. 2879–2878 (cit. on pp. 5, 31).
- [AW20] S. Apers and R. de Wolf. “Quantum Speedup for Graph Sparsification, Cut Approximation and Laplacian Solving”. In: *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 637–648 (cit. on pp. 62, 98).
- [AW99] D. S. Abrams and C. P. Williams. *Fast Quantum Algorithms for Numerical Integrals and Stochastic Processes*. [arXiv:quant-ph/9908083](https://arxiv.org/abs/quant-ph/9908083). 1999 (cit. on pp. 3, 26).
- [Bac13] F. Bach. “Learning with Submodular Functions: A Convex Optimization Perspective”. In: *Foundations and Trends in Machine Learning* 6.2-3 (2013), pp. 145–373 (cit. on pp. 103, 104).
- [BBC+01] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. “Quantum Lower Bounds by Polynomials”. In: *Journal of the ACM* 48.4 (2001), pp. 778–797 (cit. on p. 126).
- [BBHT98] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. “Tight Bounds on Quantum Searching”. In: *Fortschritte der Physik* 46.4-5 (1998), pp. 493–505 (cit. on pp. 19, 20, 143).
- [BCL13] S. Bubeck, N. Cesa-Bianchi, and G. Lugosi. “Bandits with Heavy Tail”. In: *IEEE Transactions on Information Theory* 59.11 (2013), pp. 7711–7717 (cit. on pp. 26, 28).
- [BCW98] H. Buhrman, R. Cleve, and A. Wigderson. “Quantum vs. Classical Communication and Computation”. In: *Proceedings of the 30th Symposium on Theory of Computing (STOC)*. 1998, pp. 63–68 (cit. on p. 82).
- [BCWZ99] H. Buhrman, R. Cleve, R. de Wolf, and C. Zalka. “Bounds for Small-Error and Zero-Error Quantum Algorithms”. In: *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*. 1999, pp. 358–368 (cit. on p. 41).
- [BDF+04] A. Berzina, A. Dubrovsky, R. Freivalds, L. Lace, and O. Scegulnaja. “Quantum Query Complexity for Some Graph Problems”. In: *Proceedings of the 30th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*. 2004, pp. 140–150 (cit. on p. 62).
- [BDGT11] G. Brassard, F. Dupuis, S. Gambs, and A. Tapp. *An Optimal Quantum Algorithm to Approximate the Mean and its Application for Approximating the Median of a Set of Points over an Arbitrary Distance*. [arXiv:1106.4267 \[quant-ph\]](https://arxiv.org/abs/1106.4267). 2011 (cit. on pp. 3, 25, 26, 30, 31).

Bibliography

- [Bea91] P. Beame. “A General Sequential Time-Space Tradeoff for Finding Unique Elements”. In: *SIAM Journal on Computing* 20.2 (1991), pp. 270–277 (cit. on p. 126).
- [Bel12] A. Belovs. “Span Programs for Functions with Constant-Sized 1-Certificates: Extended Abstract”. In: *Proceedings of the 44th Symposium on Theory of Computing (STOC)*. 2012, pp. 77–84 (cit. on p. 62).
- [Bel19] A. Belovs. “Quantum Algorithms for Classical Probability Distributions”. In: *Proceedings of the 27th European Symposium on Algorithms (ESA)*. 2019, 16:1–16:11 (cit. on p. 30).
- [Ben73] C. H. Bennett. “Logical Reversibility of Computation”. In: *IBM Journal of Research and Development* 17.6 (1973), pp. 525–532 (cit. on pp. 7, 18, 68, 115, 118).
- [Ben89] C. H. Bennett. “Time/Space Trade-Offs for Reversible Computation”. In: *SIAM Journal on Computing* 18.4 (1989), pp. 766–776 (cit. on pp. 18, 115, 118, 122).
- [Ber05] D. J. Bernstein. *Understanding Brute Force*. ECRYPT STVL Workshop on Symmetric Key Encryption. 2005 (cit. on p. 125).
- [Ber09] D. J. Bernstein. “Cost Analysis of Hash Collisions: Will Quantum Computers Make SHARCS Obsolete?” In: *Proceedings of the 4th Workshop on Special-purpose Hardware for Attacking Cryptographic Systems (SHARCS)*. 2009, pp. 105–116 (cit. on p. 125).
- [Ber14] D. W. Berry. “High-Order Quantum Algorithm for Solving Linear Differential Equations”. In: *Journal of Physics A: Mathematical and Theoretical* 47.10 (2014), p. 105301 (cit. on p. 87).
- [BFK+81] A. Borodin, M. J. Fischer, D. G. Kirkpatrick, N. A. Lynch, and M. Tompa. “A Time-Space Tradeoff for Sorting on Non-Oblivious Machines”. In: *Journal of Computer and System Sciences* 22.3 (1981), pp. 351–364 (cit. on pp. 125, 126, 129, 140).
- [BFM+87] A. Borodin, F. E. Fich, F. Meyer auf der Heide, E. Upfal, and A. Wigderson. “A Time-Space Tradeoff for Element Distinctness”. In: *SIAM Journal on Computing* 16.1 (1987), pp. 97–99 (cit. on pp. 126, 144).
- [BFS87] P. Bratley, B. L. Fox, and L. E. Schrage. *A Guide to Simulation*. Second. Springer-Verlag, 1987 (cit. on p. 88).
- [BGK+18] M. Braverman, A. Garg, Y. K. Ko, J. Mao, and D. Touchette. “Near-Optimal Bounds on the Bounded-Round Quantum Communication Complexity of Disjointness”. In: *SIAM Journal on Computing* 47.6 (2018), pp. 2277–2314 (cit. on p. 124).
- [BHH+19] N. Bindel, M. Hamburg, K. Hövelmanns, A. Hülsing, and E. Persichetti. “Tighter Proofs of CCA Security in the Quantum Random Oracle Model”. In: *Proceedings of the 17th Conference on Theory of Cryptography (TCC)*. 2019, pp. 61–90 (cit. on p. 126).
- [BHH11] S. Bravyi, A. W. Harrow, and A. Hassidim. “Quantum Algorithms for Testing Properties of Distributions”. In: *IEEE Transactions on Information Theory* 57.6 (2011), pp. 3971–3981 (cit. on p. 31).

- [BHMT02] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. “Quantum Amplitude Amplification and Estimation”. In: *Contemporary Mathematics* 305 (2002), pp. 53–74 (cit. on pp. 18–20, 25–27, 34).
- [BHT98a] G. Brassard, P. Høyer, and A. Tapp. “Quantum Counting”. In: *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP)*. 1998, pp. 820–831 (cit. on p. 3).
- [BHT98b] G. Brassard, P. Høyer, and A. Tapp. “Quantum Cryptanalysis of Hash and Claw-Free Functions”. In: *Proceedings of the 3rd Latin American Symposium on Theoretical Informatics (LATIN)*. 1998, pp. 163–169 (cit. on pp. 7, 125, 126).
- [Bic65] P. J. Bickel. “On Some Robust Estimates of Location”. In: *The Annals of Mathematical Statistics* 36.3 (1965), pp. 847–858 (cit. on p. 28).
- [BIJK18] A. Bhattacharya, D. Issac, R. Jaiswal, and A. Kumar. “Sampling in Space Restricted Settings”. In: *Algorithmica* 80.5 (2018), pp. 1439–1458 (cit. on p. 88).
- [BJ99] N. H. Bshouty and J. C. Jackson. “Learning DNF over the Uniform Distribution Using a Quantum Example Oracle”. In: *SIAM Journal on Computing* 28.3 (1999), pp. 1136–1153 (cit. on pp. 5, 31).
- [BJKS04] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. “An Information Statistics Approach to Data Stream and Communication Complexity”. In: *Journal of Computer and System Sciences* 68.4 (2004), pp. 702–732 (cit. on p. 116).
- [BKL+19] F. G. S. L. Brandão, A. Kaley, T. Li, C. Y.-Y. Lin, K. M. Svore, and X. Wu. “Quantum SDP Solvers: Large Speed-Ups, Optimality, and Applications to Quantum Learning”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 27:1–27:14 (cit. on p. 98).
- [BL13] K. Bringmann and K. G. Larsen. “Succinct Sampling from Discrete Distributions”. In: *Proceedings of the 45th Symposium on Theory of Computing (STOC)*. 2013, pp. 775–782 (cit. on p. 88).
- [BLM13] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013 (cit. on p. 12).
- [BP17] K. Bringmann and K. Panagiotou. “Efficient Sampling Methods for Discrete Distributions”. In: *Algorithmica* 79.2 (2017), pp. 484–508 (cit. on p. 88).
- [BR12] A. Belovs and B. W. Reichardt. “Span Programs and Quantum Algorithms for st-Connectivity and Claw Detection”. In: *Proceedings of the 20th European Symposium on Algorithms (ESA)*. 2012, pp. 193–204 (cit. on p. 62).
- [BS17] F. G. S. L. Brandão and K. M. Svore. “Quantum Speed-Ups for Solving Semidefinite Programs”. In: *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 415–426 (cit. on p. 98).
- [BSSV03] P. Beame, M. Saks, X. Sun, and E. Vee. “Time-Space Trade-off Lower Bounds for Randomized Computation of Decision Problems”. In: *Journal of the ACM* 50.2 (2003), pp. 154–195 (cit. on pp. 126, 140).

Bibliography

- [BV97] E. Bernstein and U. V. Vazirani. “Quantum Complexity Theory”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1411–1473 (cit. on p. 2).
- [BW02] H. Buhrman and R. de Wolf. “Complexity Measures and Decision Tree Complexity: A Survey”. In: *Theoretical Computer Science* 288.1 (2002), pp. 21–43 (cit. on p. 130).
- [BWP+17] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. “Quantum Machine Learning”. In: *Nature* 549.7671 (2017), pp. 195–202 (cit. on p. 97).
- [Cat12] O. Catoni. “Challenging the Empirical Mean and Empirical Variance: A Deviation Study”. In: *Annales de l’Institut Henri Poincaré, Probabilités et Statistiques* 48.4 (2012), pp. 1148–1185 (cit. on pp. 26, 43).
- [CC17] A. Chakrabarti and Y. Chen. *Time-Space Tradeoffs for the Memory Game*. [arXiv:1712.01330](https://arxiv.org/abs/1712.01330) [cs.CC]. 2017 (cit. on p. 126).
- [CCH+19] S. Chakrabarti, A. M. Childs, S.-H. Hung, T. Li, C. Wang, and X. Wu. *Quantum Algorithm for Estimating Volumes of Convex Bodies*. [arXiv:1908.03903](https://arxiv.org/abs/1908.03903) [quant-ph]. 2019 (cit. on p. 43).
- [CCKM13] A. Chakrabarti, G. Cormode, R. Kondapally, and A. McGregor. “Information Cost Tradeoffs for Augmented Index and Streaming Language Recognition”. In: *SIAM Journal on Computing* 42.1 (2013), pp. 61–83 (cit. on p. 116).
- [CCLW20] S. Chakrabarti, A. M. Childs, T. Li, and X. Wu. “Quantum Algorithms and Lower Bounds for Convex Optimization”. In: *Quantum* 4 (2020), p. 221 (cit. on p. 98).
- [CEG95] R. Canetti, G. Even, and O. Goldreich. “Lower Bounds for Sampling Algorithms for Estimating the Average”. In: *Information Processing Letters* 53.1 (1995), pp. 17–25 (cit. on p. 27).
- [CFMW10] S. Chakraborty, E. Fischer, A. Matsliah, and R. de Wolf. “New Results on Quantum Property Testing”. In: *Proceedings of the 30th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 2010, pp. 145–156 (cit. on p. 31).
- [CGJ19] S. Chakraborty, A. Gilyén, and S. Jeffery. “The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 33:1–33:14 (cit. on pp. 46, 49, 54, 62).
- [CGLQ20] K.-M. Chung, S. Guo, Q. Liu, and L. Qian. “Tight Quantum Time-Space Tradeoffs for Function Inversion”. In: *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 673–684 (cit. on p. 126).
- [CHI+18] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig. “Quantum Machine Learning: A Classical Perspective”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (2018), p. 20170551 (cit. on p. 97).
- [Chi17] A. M. Childs. *Lecture Notes on Quantum Algorithms*. Available at <https://www.cs.umd.edu/~amchilds/qa/>. 2017 (cit. on p. 15).

- [CHL21] A. M. Childs, S.-H. Hung, and T. Li. *Quantum Query Complexity with Matrix-Vector Products*. [arXiv:2102.11349](#) [quant-ph]. 2021 (cit. on p. 112).
- [CHW12] K. L. Clarkson, E. Hazan, and D. P. Woodruff. “Sublinear Optimization for Machine Learning”. In: *Journal of the ACM* 59.5 (2012), pp. 1–49 (cit. on p. 98).
- [CJ19] G. Cormode and H. Jowhari. “Lp Samplers and Their Applications: A Survey”. In: *ACM Computing Surveys* 52.1 (2019), 16:1–16:31 (cit. on pp. 116, 117).
- [CK12] A. M. Childs and R. Kothari. “Quantum Query Complexity of Minor-Closed Graph Properties”. In: *SIAM Journal on Computing* 41.6 (2012), pp. 1426–1450 (cit. on p. 62).
- [CKS17] A. M. Childs, R. Kothari, and R. D. Somma. “Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision”. In: *SIAM Journal on Computing* 46.6 (2017), pp. 1920–1950 (cit. on pp. 46, 49).
- [CLM20] T. Carlette, M. Laurière, and F. Magniez. “Extended Learning Graphs for Triangle Finding”. In: *Algorithmica* 82.4 (2020), pp. 980–1005 (cit. on p. 62).
- [CLQ20] K.-M. Chung, T.-N. Liao, and L. Qian. “Lower Bounds for Function Inversion with Quantum Advice”. In: *Proceedings of the 1st Conference on Information-Theoretic Cryptography (ITC)*. 2020, 8:1–8:15 (cit. on p. 126).
- [CLSW17] D. Chakrabarty, Y. T. Lee, A. Sidford, and S. C.-w. Wong. “Subquadratic Submodular Function Minimization”. In: *Proceedings of the 49th Symposium on Theory of Computing (STOC)*. 2017, pp. 1220–1231 (cit. on pp. 6, 98, 102–106).
- [CMS19] A. Chiesa, P. Manohar, and N. Spooner. “Succinct Arguments in the Quantum Random Oracle Model”. In: *Proceedings of the 17th Conference on Theory of Cryptography (TCC)*. 2019, pp. 1–29 (cit. on pp. 126, 128).
- [CMSZ19] J. Czejkowski, C. Majenz, C. Schaffner, and S. Zú. *Quantum Lazy Sampling and Game-Playing Proofs for Quantum Indifferentiability*. [arXiv:1904.11477v1](#) [quant-ph]. 2019 (cit. on pp. 126, 128).
- [CNS17] A. Chailloux, M. Naya-Plasencia, and A. Schrottenloher. “An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography”. In: *Proceedings of the 23th International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT)*. 2017, pp. 211–240 (cit. on p. 125).
- [CRT05] B. Chazelle, R. Rubinfeld, and L. Trevisan. “Approximating the Minimum Spanning Tree Weight in Sublinear Time”. In: *SIAM Journal on Computing* 34.6 (2005), pp. 1370–1379 (cit. on pp. 62, 83).
- [Cun85] W. H. Cunningham. “On Submodular Function Minimization”. In: *Combinatorica* 5.3 (1985), pp. 185–192 (cit. on p. 102).
- [DDKS12] I. Dinur, O. Dunkelman, N. Keller, and A. Shamir. “Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems”. In: *Proceedings of the 32th International Cryptology Conference (CRYPTO)*. 2012, pp. 719–740 (cit. on p. 125).

Bibliography

- [DEM19] C. Delaplace, A. Esser, and A. May. “Improved Low-Memory Subset Sum and LPN Algorithms via Multiple Collisions”. In: *Proceedings of the 17th IMA International Conference on Cryptography and Coding (IMACC)*. 2019, pp. 178–199 (cit. on p. 125).
- [Deu85] D. E. Deutsch. “Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer”. In: *Proceedings of the Royal Society of London Series A* 400.1818 (1985), pp. 97–117 (cit. on p. 1).
- [Deu89] D. E. Deutsch. “Quantum Computational Networks”. In: *Proceedings of the Royal Society of London Series A* 425.1868 (1989), pp. 73–90 (cit. on pp. 1, 15).
- [Dev86] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986 (cit. on p. 88).
- [DF91] M. E. Dyer and A. Frieze. “Computing the Volume of Convex Bodies: A Case where Randomness Provably Helps”. In: *Proceedings of the Symposium on Probabilistic Combinatorics and Its Applications*. 1991, pp. 123–170 (cit. on p. 3).
- [DFK91] M. Dyer, A. Frieze, and R. Kannan. “A Random Polynomial-Time Algorithm for Approximating the Volume of Convex Bodies”. In: *Journal of the ACM* 38.1 (1991), pp. 1–17 (cit. on p. 3).
- [DH96] C. Dürr and P. Høyer. *A Quantum Algorithm for Finding the Minimum*. [arXiv:quant-ph/9607014](https://arxiv.org/abs/quant-ph/9607014). 1996 (cit. on pp. 3, 28, 31, 32, 90).
- [DHHM06] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. “Quantum Query Complexity of Some Graph Problems”. In: *SIAM Journal on Computing* 35.6 (2006), pp. 1310–1328 (cit. on pp. 3, 62, 91, 98).
- [Din20] I. Dinur. “Tight Time-Space Lower Bounds for Finding Multiple Collision Pairs and Their Applications”. In: *Proceedings of the 39th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 2020, pp. 405–434 (cit. on pp. 125, 126, 128).
- [DJ92] D. Deutsch and R. Jozsa. “Rapid Solution of Problems by Quantum Computation”. In: *Proceedings of the Royal Society of London Series A* 439.1907 (1992), pp. 553–558 (cit. on p. 2).
- [DKLR00] P. Dagum, R. Karp, M. Luby, and S. Ross. “An Optimal Algorithm for Monte Carlo Estimation”. In: *SIAM Journal on Computing* 29.5 (2000), pp. 1484–1496 (cit. on pp. 26, 27, 29, 37).
- [DKW19] K. DeLorenzo, S. Kimmel, and R. T. Witter. “Applications of the Quantum Algorithm for st-Connectivity”. In: *Proceedings of the 14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*. 2019, 6:1–6:14 (cit. on p. 62).
- [DL18] H. Dell and J. Lapinskas. “Fine-Grained Reductions from Approximate Counting to Decision”. In: *Proceedings of the 50th Symposium on Theory of Computing (STOC)*. 2018, pp. 281–288 (cit. on p. 62).
- [DLLO16] L. Devroye, M. Lerasle, G. Lugosi, and R. I. Oliveira. “Sub-Gaussian Mean Estimators”. In: *The Annals of Statistics* 44.6 (2016), pp. 2695–2725 (cit. on p. 26).

- [DLM20] H. Dell, J. Lapinskas, and K. Meeks. “Approximately Counting and Sampling Small Witnesses Using a Colourful Decision Oracle”. In: *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*. 2020, pp. 2201–2180 (cit. on p. 62).
- [Duc18] J. C. Duchi. “Introductory Lectures on Stochastic Optimization”. In: *The Mathematics of Data*. Vol. 25. IAS/Park City Mathematics Series. American Mathematical Society, 2018, pp. 99–185 (cit. on p. 104).
- [Eck87] R. Eckhardt. “Stan Ulam, John von Neumann, and the Monte Carlo Method”. In: *Los Alamos Science* 15.30 (1987), pp. 131–136 (cit. on p. 1).
- [ELR15] T. Eden, A. Levi, and D. Ron. *Approximately Counting Triangles in Sublinear Time*. **ECCC Report: TR15-046**. 2015 (cit. on pp. 62, 63, 66, 70, 83).
- [ELRS17] T. Eden, A. Levi, D. Ron, and C. Seshadhri. “Approximately Counting Triangles in Sublinear Time”. In: *SIAM Journal on Computing* 46.5 (2017), pp. 1603–1646 (cit. on pp. 4, 61–64, 66, 71, 73, 76, 83).
- [ER18] T. Eden and W. Rosenbaum. “Lower Bounds for Approximating Graph Parameters via Communication Complexity”. In: *Proceedings of the Workshop on Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques (APPROX/RANDOM)*. 2018, 11:1–11:18 (cit. on pp. 81, 82).
- [ERR20] T. Eden, D. Ron, and W. Rosenbaum. *Almost Optimal Bounds for Sublinear-Time Sampling of k -Cliques: Sampling Cliques is Harder Than Counting*. **arXiv:2012.04090 [cs.DS]**. 2020 (cit. on p. 62).
- [ERS20a] T. Eden, D. Ron, and C. Seshadhri. “Faster Sublinear Approximation of the Number of k -Cliques in Low-Arboricity Graphs”. In: *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*. 2020, pp. 1467–1478 (cit. on pp. 61, 62, 83).
- [ERS20b] T. Eden, D. Ron, and C. Seshadhri. “On Approximating the Number of k -Cliques in Sublinear Time”. In: *SIAM Journal on Computing* 49.4 (2020), pp. 747–771 (cit. on pp. 61, 62, 83).
- [Fei06] U. Feige. “On Sums of Independent Random Variables with Unbounded Variance and Estimating the Average Degree in a Graph”. In: *SIAM Journal on Computing* 35.4 (2006), pp. 964–984 (cit. on pp. 3, 61, 62).
- [Fey82] R. P. Feynman. “Simulating Physics with Computers”. In: *International Journal of Theoretical Physics* 21.6/7 (1982) (cit. on p. 1).
- [Fey86] R. P. Feynman. “Quantum Mechanical Computers”. In: *Foundations of Physics* 16.6 (1986), pp. 507–531 (cit. on p. 1).
- [FGP20] H. Fichtenberger, M. Gao, and P. Peng. “Sampling Arbitrary Subgraphs Exactly Uniformly in Sublinear Time”. In: *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2020, 45:1–45:13 (cit. on p. 62).
- [Fla85] P. Flajolet. “Approximate Counting: A Detailed Analysis”. In: *BIT* 25.1 (1985), pp. 113–134 (cit. on pp. 6, 116).

- [FM85] P. Flajolet and G. N. Martin. “Probabilistic Counting Algorithms for Data Base Applications”. In: *Journal of Computer and System Sciences* 31.2 (1985), pp. 182–209 (cit. on p. 116).
- [FO89] P. Flajolet and A. M. Odlyzko. “Random Mapping Statistics”. In: *Proceedings of the 7th Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*. 1989, pp. 329–354 (cit. on pp. 6, 127, 142).
- [FS97] Y. Freund and R. E. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139 (cit. on pp. 6, 98–100).
- [Gal09] F. Le Gall. “Exponential Separation of Quantum and Classical Online Space Complexity”. In: *Theory of Computing Systems* 45.2 (2009), pp. 188–202 (cit. on pp. 7, 115, 116).
- [Gal14] F. Le Gall. “Improved Quantum Algorithm for Triangle Finding via Combinatorial Arguments”. In: *Proceedings of the 55th Symposium on Foundations of Computer Science (FOCS)*. 2014, pp. 216–225 (cit. on pp. 46, 62).
- [Gan15] S. Ganguly. “Taylor Polynomial Estimator for Estimating Frequency Moments”. In: *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*. 2015, pp. 542–553 (cit. on pp. 116, 124).
- [GAW19] A. Gilyén, S. Arunachalam, and N. Wiebe. “Optimizing Quantum Optimization Algorithms via Faster Quantum Gradient Computation”. In: *Proceedings of the 30th Symposium on Discrete Algorithms (SODA)*. 2019, pp. 1425–1444 (cit. on p. 98).
- [GK95] M. D. Grigoriadis and L. G. Khachiyan. “A Sublinear-Time Randomized Approximation Algorithm for Matrix Games”. In: *Operations Research Letters* 18.2 (1995), pp. 53–58 (cit. on p. 98).
- [GKK+09] D. Gavinsky, J. Kempe, I. Kerenidis, R. Raz, and R. de Wolf. “Exponential Separation for One-Way Quantum Communication Complexity, with Applications to Cryptography”. In: *SIAM Journal on Computing* 38.5 (2009), pp. 1695–1708 (cit. on pp. 115, 116).
- [GKNS20] A. Garg, R. Kothari, P. Netrapalli, and S. Sherif. *No Quantum Speedup over Gradient Descent for Non-Smooth Convex Optimization*. [arXiv:2010.01801](https://arxiv.org/abs/2010.01801) [cs.DS]. 2020 (cit. on p. 98).
- [GL20] A. Gilyén and T. Li. “Distributional Property Testing in a Quantum World”. In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference, (ITCS)*. 2020, 25:1–25:19 (cit. on p. 30).
- [GLS81] M. Grötschel, L. Lovász, and A. Schrijver. “The Ellipsoid Method and its Consequences in Combinatorial Optimization”. In: *Combinatorica* 1.2 (1981), pp. 169–197 (cit. on p. 102).
- [GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Vol. 2. Algorithms and Combinatorics. Springer, 1988 (cit. on p. 102).
- [GN17] F. Le Gall and S. Nakajima. “Quantum Algorithm for Triangle Finding in Sparse Graphs”. In: *Algorithmica* 79.3 (2017), pp. 941–959 (cit. on pp. 46, 62).

- [GNP13] L. Gajek, W. Niemirow, and P. Pokarowski. “Optimal Monte Carlo Integration with Fixed Relative Precision”. In: *Journal of Complexity* 29.1 (2013), pp. 4–26 (cit. on p. 27).
- [Gol17] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017 (cit. on pp. 62, 67).
- [GR02] L. K. Grover and T. Rudolph. *Creating Superpositions that Correspond to Efficiently Integrable Probability Distributions*. [arXiv:quant-ph/0208112](#). 2002 (cit. on pp. 88, 91).
- [GR08] O. Goldreich and D. Ron. “Approximating Average Parameters of Graphs”. In: *Random Structures & Algorithms* 32.4 (2008), pp. 473–493 (cit. on pp. 61, 62, 66).
- [Gro00a] L. K. Grover. “Rapid Sampling through Quantum Computing”. In: *Proceedings of the 32nd Symposium on Theory of Computing (STOC)*. 2000, pp. 618–626 (cit. on p. 88).
- [Gro00b] L. K. Grover. “Synthesis of Quantum Superpositions by Quantum Computation”. In: *Physical Review Letters* 85.6 (2000), pp. 1334–1337 (cit. on pp. 5, 87–91).
- [Gro96a] L. K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proceedings of the 28th Symposium on Theory of Computing (STOC)*. 1996, pp. 212–219 (cit. on pp. 2, 18).
- [Gro96b] L. K. Grover. *A Fast Quantum Mechanical Algorithm for Estimating the Median*. [arXiv:quant-ph/9607024](#). 1996 (cit. on p. 3).
- [Gro98] L. K. Grover. “A Framework for Fast Quantum Mechanical Algorithms”. In: *Proceedings of the 30th Symposium on Theory of Computing (STOC)*. 1998, pp. 53–62 (cit. on pp. 3, 26, 31).
- [GRS11] M. Gonen, D. Ron, and Y. Shavitt. “Counting Stars and Other Small Subgraphs in Sublinear-Time”. In: *SIAM Journal on Discrete Mathematics* 25.3 (2011), pp. 1365–1411 (cit. on pp. 61, 62, 66).
- [GS78] L. J. Guibas and R. Sedgewick. “A Dichromatic Framework for Balanced Trees”. In: *Proceedings of the 19th Symposium on Foundations of Computer Science (FOCS)*. 1978, pp. 8–21 (cit. on p. 105).
- [Ham21] Y. Hamoudi. “Quantum Sub-Gaussian Mean Estimator”. In submission. 2021 (cit. on p. 25).
- [Har08] N. J. A. Harvey. “Matchings, Matroids and Submodular Functions”. PhD thesis. Massachusetts Institute of Technology, 2008 (cit. on p. 112).
- [Has70] W. K. Hastings. “Monte Carlo Sampling Methods using Markov Chains and their Applications”. In: *Biometrika* 57.1 (1970), pp. 97–109 (cit. on p. 4).
- [HB14] T. Homem-de-Mello and G. Bayraksan. “Monte Carlo Sampling-Based Methods for Stochastic Optimization”. In: *Surveys in Operations Research and Management Science* 19.1 (2014), pp. 56–85 (cit. on p. 5).
- [Hei02] S. Heinrich. “Quantum Summation with an Application to Integration”. In: *Journal of Complexity* 18.1 (2002), pp. 1–50 (cit. on pp. 3, 25, 26, 28, 31).

- [Hei03] S. Heinrich. “From Monte Carlo to Quantum Computation”. In: *Mathematics and Computers in Simulation* 62.3–6 (2003), pp. 219–230 (cit. on p. 26).
- [Hel69] C. W. Helstrom. “Quantum Detection and Estimation Theory”. In: *Journal of Statistical Physics* 1.2 (1969), pp. 231–252 (cit. on p. 42).
- [Hel80] M. Hellman. “A Cryptanalytic Time-Memory Trade-Off”. In: *IEEE Transactions on Information Theory* 26.4 (1980), pp. 401–406 (cit. on p. 6).
- [HHL09] A. W. Harrow, A. Hassidim, and S. Lloyd. “Quantum Algorithm for Linear Systems of Equations”. In: *Physical Review Letters* 103.15 (2009), p. 150502 (cit. on pp. 87, 97).
- [HI19] A. Hosoyamada and T. Iwata. “4-Round Luby-Rackoff Construction is a qPRP”. In: *Proceedings of the 25th International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT)*. 2019, pp. 145–174 (cit. on pp. 126, 128).
- [HK12] E. Hazan and S. Kale. “Online Submodular Minimization”. In: *Journal of Machine Learning Research* 13.1 (2012), pp. 2903–2922 (cit. on p. 103).
- [HM19] Y. Hamoudi and F. Magniez. “Quantum Chebyshev’s Inequality and Applications”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 69:1–69:16 (cit. on pp. 25–28, 30, 45, 61, 115).
- [HM21a] Y. Hamoudi and F. Magniez. “Quantum Approximate Triangle Counting”. In submission. 2021 (cit. on pp. 45, 61).
- [HM21b] Y. Hamoudi and F. Magniez. “Quantum Time-Space Tradeoff for Finding Multiple Collision Pairs”. In: *Proceedings of the 16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*. 2021 (cit. on p. 125).
- [HMM93] T. Hagerup, K. Mehlhorn, and J. I. Munro. “Maintaining Discrete Probability Distributions Optimally”. In: *Proceedings of the 20nd International Colloquium on Automata, Languages, and Programming (ICALP)*. 1993, pp. 253–264 (cit. on p. 88).
- [Hop20] S. B. Hopkins. “Mean Estimation with Sub-Gaussian Rates in Polynomial Time”. In: *The Annals of Statistics* 48.2 (2020), pp. 1193–1213 (cit. on p. 43).
- [HRRS19] Y. Hamoudi, P. Rebertost, A. Rosmanis, and M. Santha. “Quantum and Classical Algorithms for Approximate Submodular Function Minimization”. In: *Quantum Information & Computation* 19.15&16 (2019), pp. 1325–1349 (cit. on pp. 87, 94, 97, 102, 103, 111).
- [HRS18] T. Häner, M. Roetteler, and K. M. Svore. *Optimizing Quantum Circuits for Arithmetic*. [arXiv:1805.12445](https://arxiv.org/abs/1805.12445) [quant-ph]. 2018 (cit. on p. 90).
- [HSTX20] A. Hosoyamada, Y. Sasaki, S. Tani, and K. Xagawa. “Quantum Algorithm for the Multicollision Problem”. In: *Theoretical Computer Science* 842 (2020), pp. 100–117 (cit. on p. 125).
- [Hub19] M. Huber. “An Optimal (ϵ, δ) -Randomized Approximation Scheme for the Mean of Random Variables with Bounded Relative Variance”. In: *Random Structures & Algorithms* 55.2 (2019), pp. 356–370 (cit. on p. 27).

- [HW20] A. W. Harrow and A. Y. Wei. “Adaptive Quantum Simulated Annealing for Bayesian Inference and Estimating Partition Functions”. In: *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*. 2020, pp. 193–212 (cit. on pp. 3, 5, 43, 62, 88).
- [HXY19] M. Hhan, K. Xagawa, and T. Yamakawa. “Quantum Random Oracle Model with Auxiliary Input”. In: *Proceedings of the 25th International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT)*. 2019, pp. 584–614 (cit. on p. 126).
- [IFF01] S. Iwata, L. Fleischer, and S. Fujishige. “A Combinatorial Strongly Polynomial Algorithm for Minimizing Submodular Functions”. In: *Journal of the ACM* 48.4 (2001), pp. 761–777 (cit. on p. 102).
- [IJ19] T. Ito and S. Jeffery. “Approximate Span Programs”. In: *Algorithmica* 81.6 (2019), pp. 2158–2195 (cit. on p. 62).
- [IW05] P. Indyk and D. P. Woodruff. “Optimal Approximations of the Frequency Moments of Data Streams”. In: *Proceedings of the 37th Symposium on Theory of Computing (STOC)*. 2005, pp. 202–208 (cit. on p. 116).
- [IW20] A. Izdebski and R. de Wolf. *Improved Quantum Boosting*. [arXiv:2009.08360 \[quant-ph\]](#). 2020 (cit. on p. 98).
- [JB11] S. Jegelka and J. A. Bilmes. “Online Submodular Minimization for Combinatorial Structures”. In: *Proceedings of the 28th International Conference on Machine Learning (ICML)*. 2011, pp. 345–352 (cit. on p. 104).
- [JJKP18] M. Jarret, S. Jeffery, S. Kimmel, and A. Piedrafita. “Quantum Algorithms for Connectivity and Related Problems”. In: *Proceedings of the 26th European Symposium on Algorithms (ESA)*. 2018, 49:1–49:13 (cit. on p. 62).
- [JL09] A. Joux and S. Lucks. “Improved Generic Algorithms for 3-Collisions”. In: *Proceedings of the 15th International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT)*. 2009, pp. 347–363 (cit. on p. 125).
- [JN14] R. Jain and A. Nayak. “The Space Complexity of Recognizing Well-Parentthesized Expressions in the Streaming Model: The Index Function Revisited”. In: *IEEE Transactions on Information Theory* 60.10 (2014), pp. 6646–6668 (cit. on p. 116).
- [Jor05] S. P. Jordan. “Fast Quantum Algorithm for Numerical Gradient Estimation”. In: *Physical Review Letters* 95.5 (2005), p. 050501 (cit. on p. 98).
- [JRS03] R. Jain, J. Radhakrishnan, and P. Sen. “A Lower Bound for the Bounded Round Quantum Communication Complexity of Set Disjointness”. In: *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*. 2003, pp. 220–229 (cit. on pp. 116, 124).
- [JSV04] M. Jerrum, A. Sinclair, and E. Vigoda. “A Polynomial-Time Approximation Algorithm for the Permanent of a Matrix with Nonnegative Entries”. In: *Journal of the ACM* 51.4 (2004), pp. 671–697 (cit. on p. 3).
- [JVV86] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. “Random Generation of Combinatorial Structures from a Uniform Distribution”. In: *Theoretical Computer Science* 43 (1986), pp. 169–188 (cit. on pp. 13, 25).

Bibliography

- [Kah50a] H. Kahn. “Random Sampling (Monte Carlo) Techniques in Neutron Attenuation Problems, I”. In: *Nucleonics* 6.5 (1950), pp. 27–37 (cit. on p. 4).
- [Kah50b] H. Kahn. “Random Sampling (Monte Carlo) Techniques in Neutron Attenuation Problems, II”. In: *Nucleonics* 6.6 (1950), pp. 60–65 (cit. on p. 4).
- [KBTB14] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev. “Why the Monte Carlo Method is so Important Today”. In: *WIREs Computational Statistics* 6.6 (2014), pp. 386–392 (cit. on p. 1).
- [KKR04] T. Kaufman, M. Krivelevich, and D. Ron. “Tight Bounds for Testing Bipartiteness in General Graphs”. In: *SIAM Journal on Computing* 33.6 (2004), pp. 1441–1483 (cit. on pp. 62, 67).
- [KL83] R. M. Karp and M. Luby. “Monte-Carlo Algorithms for Enumeration and Reliability Problems”. In: *Proceedings of the 24th Symposium on Foundations of Computer Science (FOCS)*. 1983, pp. 56–64 (cit. on p. 3).
- [KLM07] P. Kaye, R. Laflamme, and M. Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2007 (cit. on p. 15).
- [KLM89] R. M. Karp, M. Luby, and N. Madras. “Monte-Carlo Approximation Algorithms for Enumeration Problems”. In: *Journal of Algorithms* 10.3 (1989), pp. 429–448 (cit. on p. 3).
- [KM01] P. Kaye and M. Mosca. “Quantum Networks for Generating Arbitrary Quantum States”. In: *Proceedings of the International Conference on Quantum Information (ICQI)*. 2001, PB28 (cit. on pp. 88, 91, 95).
- [Knu98] D. E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*. Third. Addison-Wesley, 1998 (cit. on p. 88).
- [KNW10] D. M. Kane, J. Nelson, and D. P. Woodruff. “On the Exact Space Complexity of Sketching and Streaming Small Norms”. In: *Proceedings of the 21st Symposium on Discrete Algorithms (SODA)*. 2010, pp. 1161–1178 (cit. on pp. 123, 124).
- [KP79] R. A. Kronmal and A. V. Peterson. “On the Alias Method for Generating Random Variables from a Discrete Distribution”. In: *The American Statistician* 33.4 (1979), pp. 214–218 (cit. on pp. 88, 90, 94).
- [KŠW07] H. Klauck, R. Špalek, and R. de Wolf. “Quantum and Classical Strong Direct Product Theorems and Optimal Time-Space Tradeoffs”. In: *SIAM Journal on Computing* 36.5 (2007), pp. 1472–1493 (cit. on pp. 7, 95, 125, 126, 128, 129, 131, 140, 144).
- [KW09] A. Kitaev and W. A. Webb. *Wavefunction Preparation and Resampling using a Quantum Computer*. [arXiv:0801.0342](https://arxiv.org/abs/0801.0342) [quant-ph]. 2009 (cit. on p. 88).
- [KY76] D. E. Knuth and A. C.-C. Yao. “The Complexity of Nonuniform Random Number Generation”. In: *Proceedings of the Symposium on Algorithms and Complexity: New Directions and Recent Results*. 1976, pp. 357–428 (cit. on p. 88).
- [LCW19] T. Li, S. Chakrabarti, and X. Wu. “Sublinear Quantum Algorithms for Training Linear and Kernel-Based Classifiers”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019, pp. 3815–3824 (cit. on p. 98).

- [LM19] G. Lugosi and S. Mendelson. “Mean Estimation and Regression Under Heavy-Tailed Distributions: A Survey”. In: *Foundations of Computational Mathematics* 19.5 (2019), pp. 1145–1190 (cit. on pp. 25, 26, 28, 43).
- [LMS12] T. Lee, F. Magniez, and M. Santha. “Learning Graph Based Quantum Query Algorithms for Finding Constant-Size Subgraphs”. In: *Chicago Journal of Theoretical Computer Science* 2012.10 (2012) (cit. on p. 62).
- [LMS17] T. Lee, F. Magniez, and M. Santha. “Improved Quantum Query Algorithms for Triangle Detection and Associativity Testing”. In: *Algorithmica* 77.2 (2017), pp. 459–486 (cit. on p. 62).
- [LMT00] K.-J. Lange, P. McKenzie, and A. Tapp. “Reversible Space Equals Deterministic Space”. In: *Journal of Computer and System Sciences* 60.2 (2000), pp. 354–367 (cit. on p. 18).
- [LNW14] Y. Li, H. L. Nguyen, and D. P. Woodruff. “Turnstile Streaming Algorithms Might as Well Be Linear Sketches”. In: *Proceedings of the 46th Symposium on Theory of Computing (STOC)*. 2014, pp. 174–183 (cit. on pp. 115, 121).
- [Lov82] L. Lovász. “Submodular Functions and Convexity”. In: *Proceedings of the 11th International Symposium on Mathematical Programming (ISMP)*. 1982, pp. 235–257 (cit. on pp. 98, 102, 104).
- [LR13] T. Lee and J. Roland. “A Strong Direct Product Theorem for Quantum Query Complexity”. In: *Computational Complexity* 22.2 (2013), pp. 429–462 (cit. on p. 126).
- [LR14] R. J. Lipton and K. W. Regan. *Quantum Algorithms via Linear Algebra: A Primer*. The MIT Press, 2014 (cit. on p. 15).
- [LS90] R. Y. Levine and A. T. Sherman. “A Note on Bennett’s Time-Space Tradeoff for Reversible Computation”. In: *SIAM Journal on Computing* 19.4 (1990), pp. 673–677 (cit. on p. 18).
- [LSW15] Y. T. Lee, A. Sidford, and S. C.-w. Wong. “A Faster Cutting Plane Method and its Implications for Combinatorial and Convex Optimization”. In: *Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS)*. 2015, pp. 1049–1065 (cit. on pp. 98, 102, 112).
- [LSZ21] T. Lee, M. Santha, and S. Zhang. “Quantum Algorithms for Graph Problems with Cut Queries”. In: *Proceedings of the 32nd Symposium on Discrete Algorithms (SODA)*. 2021, pp. 939–958 (cit. on p. 112).
- [LV20] J. C. H. Lee and P. Valiant. *Optimal Sub-Gaussian Mean Estimation in \mathbb{R}* . [arXiv:2011.08384](https://arxiv.org/abs/2011.08384) [math.ST]. 2020 (cit. on pp. 26, 43).
- [LW13] Y. Li and D. P. Woodruff. “A Tight Lower Bound for High Frequency Moment Estimation with Small Error”. In: *Proceedings of the Workshop on Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques (APPROX/RANDOM)*. 2013, pp. 623–638 (cit. on pp. 116, 124).
- [LW19] T. Li and X. Wu. “Quantum Query Complexity of Entropy Estimation”. In: *IEEE Transactions on Information Theory* 65.5 (2019), pp. 2899–2921 (cit. on pp. 31, 43, 116).

Bibliography

- [LWCW21] T. Li, C. Wang, S. Chakrabarti, and X. Wu. “Sublinear Classical and Quantum Algorithms for General Matrix Games”. In: *Proceedings of the 35th Conference on Artificial Intelligence (AAAI)*. 2021 (cit. on p. 98).
- [LYC14] G. H. Low, T. J. Yoder, and I. L. Chuang. “Quantum Inference on Bayesian Networks”. In: *Physical Review A* 89 (2014), p. 062315 (cit. on p. 88).
- [LZ19a] Q. Liu and M. Zhandry. “On Finding Quantum Multi-collisions”. In: *Proceedings of the 38th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 2019, pp. 189–218 (cit. on pp. 7, 125, 126, 129).
- [LZ19b] Q. Liu and M. Zhandry. “Revisiting Post-Quantum Fiat-Shamir”. In: *Proceedings of the 39th International Cryptology Conference (CRYPTO)*. 2019, pp. 326–355 (cit. on p. 126).
- [Met87] N. Metropolis. “The Beginning of the Monte Carlo Method”. In: *Los Alamos Science* 15 (1987), pp. 125–130 (cit. on p. 1).
- [MMN14] F. Magniez, C. Mathieu, and A. Nayak. “Recognizing Well-Parentthesized Expressions in the Streaming Model”. In: *SIAM Journal on Computing* 43.6 (2014), pp. 1880–1905 (cit. on p. 116).
- [MNRs11] F. Magniez, A. Nayak, J. Roland, and M. Santha. “Search via Quantum Walk”. In: *SIAM Journal on Computing* 40.1 (2011), pp. 142–164 (cit. on p. 87).
- [MNT93] Y. Mansour, N. Nisan, and P. Tiwari. “The Computational Complexity of Universal Hashing”. In: *Theoretical Computer Science* 107.1 (1993), pp. 121–133 (cit. on p. 126).
- [Mon11] A. Montanaro. “A New Exponential Separation between Quantum and Classical One-Way Communication Complexity”. In: *Quantum Information & Computation* 11.7&8 (2011), pp. 574–591 (cit. on pp. 115, 116).
- [Mon15] A. Montanaro. “Quantum Speedup of Monte Carlo Methods”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2181 (2015), p. 20150301 (cit. on pp. 3, 25–28, 30, 34, 43, 62).
- [Mon16] A. Montanaro. “The Quantum Complexity of Approximating the Frequency Moments”. In: *Quantum Information & Computation* 16.13&14 (2016), pp. 1169–1190 (cit. on pp. 115–117).
- [Mor78] R. Morris. “Counting Large Numbers of Events in Small Registers”. In: *Communications of the ACM* 21.10 (1978), pp. 840–842 (cit. on pp. 6, 116).
- [MP78] J. I. Munro and M. S. Paterson. “Selection and Sorting with Limited Storage”. In: *Proceedings of the 19th Symposium on Foundations of Computer Science (FOCS)*. 1978, pp. 253–258 (cit. on p. 6).
- [MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995 (cit. on p. 1).
- [MRR+53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. “Equation of State Calculations by Fast Computing Machines”. In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092 (cit. on p. 4).
- [MS20] A. Montanaro and C. Shao. *Quantum Algorithms for Learning a Hidden Graph and Beyond*. [arXiv:2011.08611](https://arxiv.org/abs/2011.08611) [quant-ph]. 2020 (cit. on p. 112).

- [MSA08] V. Mnih, C. Szepesvári, and J.-Y. Audibert. “Empirical Bernstein Stopping”. In: *Proceedings of the 25th International Conference on Machine Learning (ICML)*. 2008, pp. 672–679 (cit. on p. 27).
- [MSS07] F. Magniez, M. Santha, and M. Szegedy. “Quantum Algorithms for the Triangle Problem”. In: *SIAM Journal on Computing* 37.2 (2007), pp. 413–424 (cit. on p. 62).
- [MU17] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. 2nd. Cambridge University Press, 2017 (cit. on p. 1).
- [Mut05] S. Muthukrishnan. “Data Streams: Algorithms and Applications”. In: *Foundations and Trends in Theoretical Computer Science* 1.2 (2005), pp. 117–236 (cit. on p. 118).
- [MVBS05] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa. “Transformation of Quantum States Using Uniformly Controlled Rotations”. In: *Quantum Information & Computation* 5.6 (2005), pp. 467–473 (cit. on pp. 87, 88).
- [MVN03] Y. Matias, J. S. Vitter, and W.-C. Ni. “Dynamic Generation of Discrete Random Variates”. In: *Theory of Computing Systems* 36.4 (2003), pp. 329–358 (cit. on p. 88).
- [MW10] M. Monemizadeh and D. P. Woodruff. “1-pass Relative-error Lp-sampling with Applications”. In: *Proceedings of the 21st Symposium on Discrete Algorithms (SODA)*. 2010, pp. 1143–1160 (cit. on pp. 7, 116, 117, 123).
- [NABT15] A. Nayebi, S. Aaronson, A. Belovs, and L. Trevisan. “Quantum Lower Bound for Inverting a Permutation with Advice”. In: *Quantum Information & Computation* 15.11&12 (2015), pp. 901–913 (cit. on p. 126).
- [Nay99] A. Nayak. “Lower Bounds for Quantum Computation and Communication”. PhD thesis. University of California, Berkeley, 1999 (cit. on pp. 31, 42).
- [NC11] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. Cambridge University Press, 2011 (cit. on pp. 15, 17).
- [Nes12] Y. Nesterov. “Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems”. In: *SIAM Journal on Optimization* 22.2 (2012), pp. 341–362 (cit. on p. 98).
- [Neu51] J. von Neumann. “Various Techniques Used in Connection with Random Digits”. In: *Monte Carlo Method*. Vol. 12. National Bureau of Standards Applied Mathematics Series. US Government Printing Office, 1951, pp. 36–38 (cit. on pp. 4, 88).
- [NO08] H. N. Nguyen and K. Onak. “Constant-Time Approximation Algorithms via Local Improvements”. In: *Proceedings of the 49th Symposium on Foundations of Computer Science (FOCS)*. 2008, pp. 327–336 (cit. on pp. 61, 62).
- [Nov01] E. Novak. “Quantum Complexity of Integration”. In: *Journal of Complexity* 17.1 (2001), pp. 2–16 (cit. on pp. 3, 26).

Bibliography

- [NSW16] D. Needell, N. Srebro, and R. Ward. “Stochastic Gradient Descent, Weighted Dampling, and the Randomized Kaczmarz Algorithm”. In: *Mathematical Programming* 155.1-2 (2016), pp. 549–573 (cit. on pp. 97, 98).
- [NT17] A. Nayak and D. Touchette. “Augmented Index and Quantum Streaming Algorithms for DYCK(2)”. In: *Proceedings of the 32nd Computational Complexity Conference (CCC)*. 2017 (cit. on p. 116).
- [NW99] A. Nayak and F. Wu. “The Quantum Query Complexity of Approximating the Median and Related Statistics”. In: *Proceedings of the 31st Symposium on Theory of Computing (STOC)*. 1999, pp. 384–393 (cit. on pp. 3, 27, 28, 30, 31).
- [NY83] A. S. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley & Sons, 1983 (cit. on p. 25).
- [OBD18] D. Orsucci, H. J. Briegel, and V. Dunjko. “Faster Quantum Mixing for Slowly Evolving Sequences of Markov Chains”. In: *Quantum* 2 (2018), p. 105 (cit. on pp. 5, 88).
- [ODo15] R. O’Donnell. *Lecture Notes on Quantum Computation and Information*. Available at <http://www.cs.cmu.edu/~odonnell/quantum15/>. 2015 (cit. on p. 15).
- [ORR13] M. Ozols, M. Roetteler, and J. Roland. “Quantum Rejection Sampling”. In: *ACM Transactions on Computation Theory* 5.3 (2013), 11:1–11:33 (cit. on pp. 5, 88).
- [ORRR12] K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. “A Near-Optimal Sublinear-Time Algorithm for Approximating the Minimum Vertex Cover Size”. In: *Proceedings of the 23rd Symposium on Discrete Algorithms (SODA)*. 2012, pp. 1123–1131 (cit. on p. 61).
- [OW99] P. C. van Oorschot and M. J. Wiener. “Parallel Collision Search with Cryptanalytic Applications”. In: *Journal of Cryptology* 12.1 (1999), pp. 1–28 (cit. on pp. 125–128).
- [PB11] M. Plesch and v. Brukner. “Quantum-State Preparation with Universal Gate Decompositions”. In: *Physical Review A* 83 (2011), p. 032302 (cit. on pp. 87, 88).
- [Pid19] S. Piddock. *Quantum Walk Search Algorithms and Effective Resistance*. [arXiv:1912.04196](https://arxiv.org/abs/1912.04196) [quant-ph]. 2019 (cit. on p. 62).
- [Pol75] J. M. Pollard. “A Monte Carlo Method for Factorization”. In: *BIT Numerical Mathematics* 15.3 (1975), pp. 331–334 (cit. on pp. 6, 125, 126).
- [PW09] D. Poulin and P. Wocjan. “Sampling from the Thermal Quantum Gibbs State and Evaluating Partition Functions with a Quantum Computer”. In: *Physical Review Letters* 103.22 (2009), p. 220502 (cit. on p. 3).
- [Rab76] M. O. Rabin. “Probabilistic Algorithms”. In: *Proceedings of the Symposium on Algorithms and Complexity: New Directions and Recent Results*. 1976, pp. 21–39 (cit. on p. 1).
- [Raz03] A. A. Razborov. “Quantum Communication Complexity of Symmetric Predicates”. In: *Izvestiya: Mathematics* 67.1 (2003), pp. 145–159 (cit. on p. 82).

- [Raz18] R. Raz. “Fast Learning Requires Good Memory: A Time-Space Lower Bound for Parity Learning”. In: *Journal of the ACM* 66.1 (2018) (cit. on p. 6).
- [RC04] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. 2nd. Springer-Verlag New York, 2004 (cit. on p. 4).
- [RHR+21] P. Reberntrost, Y. Hamoudi, M. Ray, X. Wang, S. Yang, and M. Santha. “Quantum Algorithms for Hedging and the Learning of Ising Models”. In: *Physical Review A* 103 (2021), p. 012418 (cit. on pp. 97, 112).
- [RT14] P. Richtárik and M. Takác. “Iteration Complexity of Randomized Block-Coordinate Descent Methods for Minimizing a Composite Function”. In: *Mathematical Programming* 144.1-2 (2014), pp. 1–38 (cit. on p. 98).
- [Rus91] K. G. Russell. “Estimating the Value of e by Simulation”. In: *The American Statistician* 45.1 (1991), pp. 66–68 (cit. on p. 1).
- [SBBK08] R. D. Somma, S. Boixo, H. Barnum, and E. Knill. “Quantum Simulations of Classical Annealing Processes”. In: *Physical Review Letters* 101 (2008), p. 130504 (cit. on pp. 5, 88).
- [Sch00] A. Schrijver. “A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time”. In: *Journal of Combinatorial Theory Series B* 80.2 (2000), pp. 346–355 (cit. on p. 102).
- [Ser03] R. A. Servedio. “Smooth Boosting and Learning with Malicious Noise”. In: *Journal of Machine Learning Research* 4 (2003), pp. 633–648 (cit. on p. 98).
- [Ses15] C. Seshadhri. *A Simpler Sublinear Algorithm for Approximating the Triangle Count*. [arXiv:1505.01927](https://arxiv.org/abs/1505.01927) [cs.DS]. 2015 (cit. on pp. 62, 63, 66, 68, 71, 83).
- [Shi05] Y. Shi. “Quantum and Classical Tradeoffs”. In: *Theoretical Computer Science* 344.2 (2005), pp. 335–345 (cit. on p. 88).
- [Sho97] P. W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509 (cit. on p. 2).
- [Sim97] D. R. Simon. “On the Power of Quantum Computation”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1474–1483 (cit. on p. 2).
- [SLSB19] Y. R. Sanders, G. H. Low, A. Scherer, and D. W. Berry. “Black-Box Quantum State Preparation without Arithmetic”. In: *Physical Review Letters* 122.2 (2019), p. 020502 (cit. on pp. 88, 90, 95).
- [Špa08] R. Špalek. “The Multiplicative Quantum Adversary”. In: *Proceedings of the 23rd Computational Complexity Conference (CCC)*. 2008, pp. 237–248 (cit. on p. 126).
- [SS06] A. N. Soklakov and R. Schack. “Efficient State Preparation for a Register of Quantum Bits”. In: *Physical Review A* 73 (2006), p. 012307 (cit. on p. 88).
- [SV05] L. J. Schulman and V. V. Vazirani. “A Computationally Motivated Definition of Parametric Estimation and its Applications to the Gaussian Distribution”. In: *Combinatorica* 25.4 (2005), pp. 465–486 (cit. on p. 26).
- [SV09] T. Strohmer and R. Vershynin. “A Randomized Kaczmarz Algorithm with Exponential Convergence”. In: *Journal of Fourier Analysis and Applications* 15.2 (2009), pp. 262–278 (cit. on pp. 5, 98).

- [SYZ04] X. Sun, A. C.-C. Yao, and S. Zhang. “Graph Properties and Circular Functions: How Low Can Quantum Query Complexity Go?” In: *Proceedings of the 19th Computational Complexity Conference (CCC)*. 2004, pp. 286–293 (cit. on p. 62).
- [Sze04] M. Szegedy. “Quantum Speed-Up of Markov Chain Based Algorithms”. In: *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS)*. 2004, pp. 32–41 (cit. on p. 5).
- [Ter99] B. M. Terhal. “Quantum Algorithms and Quantum Entanglement”. PhD thesis. University of Amsterdam, 1999 (cit. on pp. 3, 25, 26, 30, 34).
- [TS13] A. Ta-Shma. “Inverting Well Conditioned Matrices in Quantum Logspace”. In: *Proceedings of the 45th Symposium on Theory of Computing (STOC)*. 2013, pp. 881–890 (cit. on p. 7).
- [TW02] J. F. Traub and H. Wozniakowski. “Path Integration on a Quantum Computer”. In: *Quantum Information Processing* 1.5 (2002), pp. 365–388 (cit. on pp. 3, 26).
- [Vos91] M. D. Vose. “A Linear Algorithm for Generating Random Numbers with a Given Distribution”. In: *IEEE Transactions on Software Engineering* 17.9 (1991), pp. 972–975 (cit. on pp. 88, 90, 94).
- [WA08] P. Wocjan and A. Abeyesinghe. “Speedup via Quantum Sampling”. In: *Physical Review A* 78.4 (2008), p. 042336 (cit. on pp. 5, 88).
- [Wag02] D. Wagner. “A Generalized Birthday Problem”. In: *Proceedings of the 22nd International Cryptology Conference (CRYPTO)*. 2002, pp. 288–304 (cit. on p. 125).
- [Wal74] A. J. Walker. “New Fast Method for Generating Discrete Random Numbers with Arbitrary Frequency Distributions”. In: *Electronics Letters* 10.8 (1974), pp. 127–128 (cit. on p. 88).
- [Wal77] A. J. Walker. “An Efficient Method for Generating Discrete Random Variables with General Distributions”. In: *ACM Transactions on Mathematical Software* 3.3 (1977), pp. 253–256 (cit. on pp. 88, 90, 94).
- [Wan17] G. Wang. “Efficient Quantum Algorithms for Analyzing Large Sparse Electrical Networks”. In: *Quantum Information & Computation* 17.11&12 (2017), pp. 987–1026 (cit. on p. 62).
- [WCNA09] P. Wocjan, C.-F. Chiang, D. Nagaj, and A. Abeyesinghe. “Quantum Algorithm for Approximating Partition Functions”. In: *Physical Review A* 80.2 (2009), p. 022340 (cit. on p. 3).
- [WG17] N. Wiebe and C. Grandade. “Can Small Quantum Systems Learn”. In: *Quantum Information & Computation* 17.7&8 (2017), pp. 568–594 (cit. on p. 88).
- [Wie04] M. J. Wiener. “The Full Cost of Cryptanalytic Attacks”. In: *Journal of Cryptology* 17.2 (2004), pp. 105–124 (cit. on p. 125).
- [Wol19] R. de Wolf. *Quantum Computing: Lecture Notes*. [arXiv:1907.09415](https://arxiv.org/abs/1907.09415) [quant-ph]. 2019 (cit. on p. 15).
- [WYLC21] D. Wang, X. You, T. Li, and A. M. Childs. “Quantum Exploration Algorithms for Multi-Armed Bandits”. In: *Proceedings of the 35th Conference on Artificial Intelligence (AAAI)*. 2021 (cit. on p. 46).

- [WZ12] D. P. Woodruff and Q. Zhang. “Tight Bounds for Distributed Functional Monitoring”. In: *Proceedings of the 44th Symposium on Theory of Computing (STOC)*. 2012, pp. 941–960 (cit. on pp. 7, 116).
- [Yao93] A. C.-C. Yao. “Quantum Circuit Complexity”. In: *Proceedings of the 34th Symposium on Foundations of Computer Science (FOCS)*. 1993, pp. 352–361 (cit. on p. 15).
- [Yao94] A. C.-C. Yao. “Near-Optimal Time-Space Tradeoff for Element Distinctness”. In: *SIAM Journal on Computing* 23.5 (1994), pp. 966–975 (cit. on pp. 126, 140, 144).
- [YYI12] Y. Yoshida, M. Yamamoto, and H. Ito. “Improved Constant-Time Approximation Algorithms for Maximum Matchings and Other Optimization Problems”. In: *SIAM Journal on Computing* 41.4 (2012), pp. 1074–1093 (cit. on pp. 61, 62).
- [Zal98] C. Zalka. “Simulating Quantum Systems on a Quantum Computer”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 313–322 (cit. on pp. 88, 91).
- [Zha15] M. Zhandry. “A Note on the Quantum Collision and Set Equality Problems”. In: *Quantum Information & Computation* 15.7&8 (2015), pp. 557–567 (cit. on pp. 126, 127).
- [Zha19] M. Zhandry. “How to Record Quantum Queries, and Applications to Quantum Indifferentiability”. In: *Proceedings of the 39th International Cryptology Conference (CRYPTO)*. 2019, pp. 239–268 (cit. on pp. 7, 126, 128–131).
- [Zhu12] Y. Zhu. “Quantum Query Complexity of Constant-Sized Subgraph Containment”. In: *International Journal of Quantum Information* 10.03 (2012), p. 1250019 (cit. on p. 62).
- [ZLL20] C. Zhang, J. Leng, and T. Li. *Quantum Algorithms for Escaping from Saddle Points*. [arXiv:2007.10253](#) [quant-ph]. 2020 (cit. on p. 98).
- [ZX17] Y. Zhang and L. Xiao. “Stochastic Primal-Dual Coordinate Method for Regularized Empirical Risk Minimization”. In: *Journal of Machine Learning Research* 18.84 (2017), pp. 1–42 (cit. on pp. 97, 98).
- [ZZ15] P. Zhao and T. Zhang. “Stochastic Optimization with Importance Sampling for Regularized Loss Minimization”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 2015, pp. 1–9 (cit. on pp. 97, 98).