Université Paris Cité

École Doctorale de Sciences Mathématiques de Paris Centre (ED 386)
Institut de Recherche en Informatique Fondamentale

Thèse de doctorat en Informatique

Quantum property testing and algebraic topology

Présentée par

Dániel Szabó

Dirigée par Frédéric Magniez

Présentée et soutenue publiquement le 09/09/2025

Université Paris Cité

École Doctorale de Sciences Mathématiques de Paris Centre (ED 386)
Centre National de la Recherche Scientifique
Institut de Recherche en Informatique Fondamentale

Thèse de doctorat en Informatique

Quantum property testing and algebraic topology

Présentée par
Dániel Szabó

Dirigée par Frédéric Magniez

Présentée et soutenue publiquement le 09/09/2025

Devant un jury composé de :

Frédéric Magniez	Directeur de recherche, CNRS Paris	Directeur de thèse
Stacey Jeffery	Professor, CWI, University of Amsterdam	Rapportrice
André Chailloux	Chargé de recherche, Inria de Paris	Rapporteur
Sophie Laplante	Professeure, Université Paris Cité	Examinatrice
Alex B. Grilo	Chargé de recherche, CNRS Paris	Examinateur
Simon Apers	Chargé de recherche, CNRS Paris	Membre invité

Acknowledgements

I am really grateful for everyone who helped me in any way during my PhD! In case I forgot to mention you, I am sorry, please don't be offended, I'm writing this last-minute.

First of all, thank you Frédéric for being a great advisor! I really appreciate your scientific and administrative help during these four years. You always managed to find the time to discuss with me, even when you were the director of IRIF. I also enjoyed spending some free time with you in restaurants, bars, or playing foosball at the airport.

Thank you, Simon, you were like a co-advisor to me. You introduced me to the world of simplicial complexes, and the first publications of my PhD came from working with you. Thank you for organising our quantum group meetings and the dinners at la Felicità!

I am grateful to the reviewers of my PhD thesis, Stacey and André, for reading and evaluating the manuscript. And to the jury members, Sophie and Alex, for contributing to my defence with their work.

I am also grateful to Frédéric and Miklós for making possible our exchange with Jinge between Paris and Singapore. Similarly, I thank Yassine for inviting me to visit LaBRI in Bordeaux. For all the other trips to conferences and other events too, I am grateful for the funding. All these missions helped me grow both scientifically and personally.

I thank my other co-authors, Sayantan and Sander for the work we have done together, as well as the collaborators I haven't mentioned yet, with whom we have had some interesting projects: Galina, Chandrima, Arjan, Roman, Matt and Luke.

I would also like to thank Kati, my former advisor in Budapest: without her guidance and recommendation I wouldn't have started this PhD with Frédéric. Köszönöm, és örülök, hogy több év után is tartjuk a kapcsolatot!

After the scientific part, I would like to thank all the people that were around me during these years. First, Enrique: we started our PhD at the same time in neighbouring offices, and we went for lunch together. Then I really wouldn't have thought that our relationship would evolve the way it did. Pero estoy muy feliz de que nos hayamos encontrado, ¡gracias por todo!

My first friend in Paris was Théo, whom I had the pleasure of meeting in Sweden a year before moving here. Merci de me permettre de pratiquer le français avec toi et d'explorer les musées de Paris ensemble.

— those who first welcomed me at IRIF for making the beginning of this journey warm and smooth: Dániel, Avi, Robin, Mikaël, Sander, Simon M., Pierre.

— my first group of friends in Paris for the many shared lunches and birthday celebrations: Enrique, Filippo, Wael, Mouna, Weiqiang.

— the climbing group, for turning sport into something entertaining and social: Enrique, Filippo, Mónika, Daniel, Mikaël, Lennart, Benjamin, Sander and Camille. A special thanks to Filippo for introducing us to bouldering.

- my roommates, for maintaining a generally calm and supportive environment for work: Wael, Roman, Elie, Benjamin, Gaëtan, Anupa and Zeinab.
- the quantum group, for the enlightening quantum meetings and for the company at several conferences and workshops: Frédéric, Simon, Arjan, Simona, Roman, Elie, Maël, Benjamin, Étienne, Chandrima, Isabella, Sebastian, Arthur.
- Mónika, Daniel and Mikaël for their warm friendship, and the many times we played board games, shared meals and went to see Mikaël on the stage.
- more friends from IRIF for the conversations, delicious dinners and joyful celebrations: Klara, Srinidhi, Shamisa and Lucie.
- the administration for their help and support: Inès, Jemuel, Marie-Laure, Mélissa, Maximilien, Aurore, Sandrine, Juliette and others.
- and many others whose kindness and company made me look forward to coming to the lab every day: Christoph, Alex, Nikolas, Kelsey, Kaartik, Mahshid, Junyao, Dung, Clément, Maud, Bernardo, Victor, Umberto, Soumyajit, Minh Hang, Vincent, Geoffroy.

Meeting Enrique also meant meeting his friends in Paris, this is how I entered the group of friends with Ruth, Manu, María, Quentin, Thomas, Pepito, Clara, Anaïs and Luna. Merci pour les repas, les jeux de société et les évènements culturels.

Last but not least, I would like to thank my family and my Hungarian friends. Köszönöm a kitartást, hogy sok, távol eltöltött év után is megmaradt a kapcsolat. Maradjon ez így! Külön köszönet illeti a szüleimet amiért ilyen jól viselik azt, hogy a négy gyerekük négy különböző országban él.

Abstract

This thesis explores the power and limitations of quantum computing by developing new classical and quantum algorithms, as well as establishing lower bounds. Along the way, we revisit and extend existing results to new settings. We use two types of algorithmic models: approximation algorithms and property testing.

The problems studied fall into two main categories. The first is motivated by topological data analysis and we look at it through the lens of algebraic topology. Here, the input is a simplicial complex, which models high-dimensional relationships in data. The objective is to estimate the number of high-dimensional “holes” – formally, the Betti numbers, or the ranks of the complex’s homology groups.

A prior result presented an efficient quantum algorithm for additively estimating normalised high-dimensional Betti numbers. We complement this by providing a classical benchmark: a randomised path-integral Monte Carlo algorithm. While the quantum algorithm is efficient when the desired precision and the spectral gap of the combinatorial Laplacian are inverse-polynomial, our randomised algorithm is efficient for a more restricted parameter regime – specifically, when the precision and the spectral gap are constant. For the special case of clique complexes, we provide an extension of this regime.

In property testing it suffices for the algorithm to distinguish with high probability inputs that satisfy some property from those that are “far” (according to some distance measure and parameter) from any input that satisfies it. We also investigate a property testing problem in topological data analysis: determining whether a clique complex has a large Betti number (over the finite field with two elements), or is far from any such complex. We show that a constant number of queries suffice to solve this problem for clique complexes if the dimension and the proximity parameter are constants.

The second class of problems relates to the k-collision problem, where the goal is to detect whether a string contains k-tuples of identical characters. We formalise this as a property testing problem: distinguish inputs with no k-collisions from those that are far from this property. While the classical complexity of this problem is settled, the quantum case remained open, with only partial upper and lower bounds known in this setting.

We generalise a known quantum algorithm for the k-collision problem to a subgraph-freeness property testing problem in the directed bounded-degree model. We then prove a lower bound using the dual polynomial method, extending prior results to a broader setting. Finally, we present some graph property testing problems, among them the testing version of 3-colourability, and show that they have asymptotically maximal quantum query complexity in the bounded-degree model.

Keywords: query complexity, quantum computing, algorithms, algebraic topology, topological data analysis, property testing, collision finding, dual polynomial method, bounded-degree graphs.

Résumé

Titre : Test de propriétés quantique et topologie algébrique

Cette thèse explore la puissance et les limitations du calcul quantique en développant de nouveaux algorithmes classiques et quantiques, ainsi qu'en établissant des bornes inférieures. Au passage, nous revisitons des résultats existants et les étendons à de nouveaux contextes. Nous utilisons deux types de modèles algorithmiques : les algorithmes d'approximation et les tests de propriété.

Les problèmes étudiés se répartissent en deux catégories principales. La première est motivée par l'analyse topologique des données que nous abordons sous l'angle de la topologie algébrique. Ici, l'entrée est un complexe simplicial, qui modélise des relations de haute dimension dans les données. L'objectif est d'estimer le nombre de « trous » de haute dimension – formellement, les nombres de Betti, ou les rangs des groupes d'homologie du complexe.

Un algorithme quantique efficace était déjà connu pour estimer de manière additive les nombres de Betti normalisés en haute dimension. Nous complétons ce résultat en fournissant une solution classique : un algorithme probabiliste de Monte Carlo à intégrale de chemin. Alors que l'algorithme quantique est efficace lorsque la précision souhaitée et l'écart spectral du laplacien combinatoire sont inversement polynomiaux, notre algorithme probabiliste est efficace dans un régime de paramètres plus restreint – en particulier, lorsque la précision et l'écart spectral sont constants. Pour le cas particulier des complexes de cliques, nous proposons une extension de ce régime.

Dans les tests (probabilistes) de propriétés, il suffit que l'algorithme distingue avec une forte probabilité les entrées qui satisfont une certaine propriété de celles qui sont « éloignées » (selon une certaine distance et un paramètre) de toute entrée qui la satisfait. Nous étudions également un problème de test de propriété en analyse topologique de données : déterminer si un complexe de cliques a un nombre de Betti élevé (sur le corps fini à deux éléments), ou s'il est éloigné de tout tel complexe. Nous montrons qu'un nombre constant de requêtes suffit à résoudre ce problème pour les complexes de cliques si la dimension et le paramètre de proximité sont constants.

La seconde classe de problèmes est liée au problème des k-collisions, où le but est de détecter si une chaîne de caractères contient k-uplets de caractères identiques. Nous formalisons cela comme un problème de test de propriété : distinguer les entrées sans k-collisions de celles qui en sont éloignées. Alors que la complexité classique de ce problème est établie, le cas quantique restait ouvert, avec seulement des bornes inférieures et supérieures partielles connues dans ce cadre.

Nous généralisons un algorithme quantique connu pour le problème des k-collisions à un problème de test de la propriété de non-présence de sous-graphes dans le modèle orienté à degré borné. Nous prouvons ensuite une borne inférieure en utilisant la méthode polynomiale duale, étendant des résultats antérieurs à un cadre plus général. Enfin, nous présentons quelques problèmes de test de propriétés de graphes, parmi lesquels la version test de la 3-colorabilité, et montrons qu'ils ont une complexité de requête quantique asymptotiquement maximale dans le modèle à degré borné.

Mots clefs : complexité en requête, calcul quantique, algorithmes, topologie algébrique, analyse topologique des données, test de propriété, recherche de collisions, méthode polynomiale duale, graphes à degré borné.

Contents

List of tables

List of figures

Résumé substantiel en français

De nos jours, la quantité de données numériques structurées dans nos vies augmente à un rythme effréné, ce qui rend le traitement efficace de l'information très important. Par exemple, l'ensemble de tous les articles scientifiques peut être représenté comme un gigantesque réseau orienté de nœuds (articles) reliés par des arêtes (références entre les articles), et l'on pourrait être intéressé par trouver un article influent, c'est-à-dire un article ayant plus d'un certain nombre de citations. Parmi les autres exemples de grands réseaux, on peut citer le World Wide Web, les réseaux sociaux, tous les messages d'une application de messagerie, etc. Les graphes sont d'une importance capitale pour comprendre ces grands réseaux, car ils offrent un moyen naturel de représenter et d'analyser les relations complexes au sein des ensembles de données.

En général, lorsqu'il a accès à un objet d'entrée de taille énorme, un algorithme doit résoudre un problème, par exemple décider s'il satisfait à une certaine propriété. Parfois, le simple fait de lire l'ensemble des données prendrait trop de temps. Dans ce cas, nous aimerions disposer d'un *algorithme sous-linéaire* qui résolve le problème. Plusieurs paradigmes différents visent à atteindre cet objectif. Une solution possible consiste à réduire la quantité de données par échantillonnage aléatoire : en émettant certaines hypothèses sur l'entrée et en ne considérant qu'une petite partie de celle-ci, le problème peut dans certains cas être résolu avec une probabilité élevée. Une autre possibilité consiste à utiliser des phénomènes quantiques susceptibles d'accélérer le calcul.

Vu d'un niveau élevé, dans cette thèse, nous examinons l'accélération de calcul que ces techniques peuvent apporter. D'une part, cela signifie concevoir de nouveaux algorithmes efficaces qui résolvent un problème donné ; d'autre part, prouver des bornes inférieures montrant qu'aucun algorithme qui résout le problème ne peut être plus efficace que la borne inférieure. De plus, on peut comparer l'efficacité avec laquelle une tâche peut être résolue dans différents modèles. Par exemple, l'une des questions majeures de l'informatique quantique est de trouver des problèmes utiles présentant un avantage quantique exponentiel, c'est-à-dire que les ordinateurs quantiques peuvent les résoudre beaucoup plus efficacement que les machines classiques. Par conséquent, améliorer l'efficacité classique d'un problème comme celui-ci en proposant un nouvel algorithme est également intéressant du point de vue de l'informatique quantique.

Aperçu des résultats

Voici un bref aperçu général des principales contributions de cette thèse.

Algorithmes classiques pour l'estimation des nombres de Betti

Approximation additive des nombres de Betti

Dans Chapitre 3, les résultats de deux articles sont présentés. Tout d'abord, sur la base de [**AGSS23**], nous proposons un algorithme classique pour le problème suivant. L'entrée est un complexe simplicial K avec n sommets, d_k k-simplexes et le k-ième nombre de Betti β_k. La sortie est une estimation additive ε du nombre de Betti normalisé β_k/d_k. Nous supposons que nous avons accès à l'échantillonnage et à la requête du complexe simplicial K en entrée. Nous pouvons également dire qu'en temps polynomial, nous pouvons vérifier si un ensemble de sommets est un simplexe dans K, et nous pouvons obtenir un simplexe aléatoire d'une taille donnée.

Un élément important de notre algorithme est une matrice H liée au Laplacien combinatoire du complexe K. Nous montrons que la trace normalisée d'une puissance suffisamment élevée de H donne une estimation du k-ième nombre de Betti normalisé. Nous estimons ensuite cette trace normalisée en remarquant qu'elle correspond à l'espérance d'une variable aléatoire qui peut être calculée par un processus de Monte Carlo.

Intuitivement, nous partons d'un simplexe aléatoire de K et effectuons une marche aléatoire sur les simplexes de K avec des probabilités de transition correspondant aux entrées de H. En utilisant une borne de concentration standard, nous pouvons estimer le nombre de fois où nous devons répéter ce processus afin d'obtenir une approximation suffisamment bonne du k-ième nombre de Betti normalisé.

La complexité de cet algorithme de base peut être légèrement améliorée en utilisant les polynômes de Chebyshev pour estimer la puissance de H. De plus, dans le cas particulier où K est un complexe clique, nous pouvons montrer que H est plus creuse qu'en général, ce qui réduit le nombre de répétitions nécessaires pour obtenir l'estimation souhaitée.

On savait déjà avant nos travaux que les algorithmes quantiques peuvent résoudre cette tâche efficacement, même pour de grandes valeurs de k, mais aucun algorithme classique efficace n'était connu dans ce régime. Plus précisément, la complexité temporelle de l'algorithme quantique de [LGZ16] est polynomiale en n, $1/\varepsilon$ et $1/\gamma$, où n est le nombre de sommets, ε est le paramètre de précision additive et γ est l'écart spectral du Laplacien combinatoire.

De cette manière, notre algorithme sert de référence classique pour les algorithmes quantiques, car il montre que le problème peut être résolu en temps polynomial même de manière classique, bien que pour un ensemble de paramètres plus restreint que dans le cas quantique. En particulier, pour les complexes simpliciaux généraux, notre algorithme fonctionne en temps polynomial si ε et γ sont des constantes. Dans le cas particulier des complexes de cliques, nous obtenons un résultat légèrement amélioré : par exemple, si $k \in \Omega(n)$ et γ est une constante, alors ε peut être inversement polynomial en n.

Test de propriété des très grands nombres de Betti

Dans le même chapitre, sur la base d'un autre article [**SA25**], nous examinons l'homologie simpliciale dans le cadre du test de propriétés. Ayant accès au graphe sous-jacent d'un complexe de cliques, l'algorithme doit distinguer si β_k est proche du maximum possible ou si l'entrée est à une distance ε de celui-ci. Nous montrons que la complexité de requête de cette tâche dépend uniquement du paramètre de proximité ε, c'est-à-dire qu'elle est indépendante de la taille de l'entrée.

Nous utilisons une notion d'indépendance des k-simplexes qui provient de la théorie des matroïdes, et nous désignons le nombre maximal de k-simplexes indépendants par r_k. Nous prouvons que r_k ne peut pas être beaucoup plus petit que le nombre total de k-simplexes d_k. De plus, cette notion permet d'obtenir une expression élégante de β_k qui inclut les paramètres d_k, r_k et r_{k-1}.

Nous prouvons ensuite notre résultat, d'abord dans le cas particulier de $k = 0$, puis de manière générale. Les preuves utilisent les formules mentionnées précédemment pour montrer qu'un nombre de Betti k très élevé signifie peu de $(k+1)$-simplexes indépendants, ce qui signifie également peu de $(k + 1)$-simplexes au total. Dans un complexe clique, un simplexe $(k+1)$ est une $(k+2)$-clique, donc dans ce cas, le graphe est proche de ne contenir aucune $(k + 2)$-clique. De plus, nous pouvons montrer qu'être loin d'avoir un grand β_k implique être loin de ne contenir aucune $(k + 2)$-clique. De cette manière, nous réduisons notre problème au test de propriété tolérant de l'absence de cliques de taille $(k+2)$. On sait que la complexité des requêtes ne dépend que des paramètres de proximité, en utilisant des résultats bien connus [Für95, FN07].

Test de propriétés quantiques

Nos travaux présentés dans Chapitre 4 s'appuient sur **[AMSS25]** et lancent l'étude du test de propriétés quantiques des graphes orientés dans le modèle à degré borné, où l'algorithme a accès à la liste d'adjacence des arêtes sortantes des sommets du graphe. Nous considérons un problème de test d'absence de sous-graphes qui est une généralisation de l'absence de k-étoiles pour une constante k. Une k-étoile est un graphe avec $(k + 1)$ sommets : un sommet central et k sommets extérieurs, et il existe une arête de chaque sommet extérieur vers le sommet central.

Nous rappelons l'exemple donné au début de ce résumé, où des articles scientifiques se réfèrent les uns aux autres. Cela correspond à ce modèle : lorsque nous lisons un article, nous ne voyons que les articles qu'il cite, mais nous voulons trouver un article qui est cité par de nombreux (k) autres, ce qui correspond exactement à une k-étoile.

Nous relions le problème de l'absence de sous-graphes au problème de k-collision de plusieurs manières. Tout d'abord, nous proposons un algorithme pour notre problème d'absence de sous-graphes en généralisant l'algorithme de recherche de k-collisions de [LZ19] aux graphes. De plus, nous devons prouver que cette approche fonctionne dans le contexte du test de propriété. En particulier, ils supposent que leur entrée contient de nombreuses k-collisions, mais nous devons prouver qu'une entrée qui est loin d'être sans sous-graphes contient de nombreux sous-graphes qui sont disjoints dans un certain sens approprié.

Intuitivement, pour le cas particulier des k-étoiles, notre algorithme échantillonne d'abord certains sommets et interroge leurs voisins. Ensuite, la recherche de Grover est utilisée pour rechercher d'autres sommets qui ont un voisin déjà interrogé dans leur voisinage, ce qui permet de trouver plusieurs étoiles à 2 branches. Grâce à des recherches de Grover itérées, certaines de ces étoiles deviennent de plus en plus grandes jusqu'à ce qu'une k-étoile soit trouvée.

Pour la borne inférieure, nous donnons une réduction simple du test de propriété de l'absence de k-collisions au problème de l'absence de k-étoiles. De cette façon, il suffit de donner une borne inférieure au premier problème, ou même à un cas particulier de celui-ci. En particulier, nous considérons le problème où il faut distinguer entre l'absence de k-collisions et de nombreuses valeurs distinctes où une k-collision se produit. Ce problème peut être facilement formulé comme une composition de fonctions booléennes simples (une version promesse de OU et la fonction seuil) si nous utilisons un encodage binaire approprié de l'entrée.

La méthode de borne inférieure que nous utilisons est appelée méthode polynomiale duale, dans laquelle il faut fournir un polynôme - que nous appelons le polynôme dual - qui certifie que tout algorithme quantique doit effectuer de nombreuses requêtes pour résoudre le problème. Pour certaines fonctions de base, il existe des polynômes duals connus, et il existe également des moyens de créer un polynôme dual potentiel pour une composition de ces fonctions simples. C'est pourquoi il est important d'envisager une version de notre problème que nous pouvons formuler ainsi.

La difficulté réside dans le fait que le polynôme que nous obtenons en utilisant ces éléments constitutifs n'est pas encore un bon polynôme dual, car il ne satisfait pas à l'une des contraintes qu'il devrait satisfaire. De plus, le respect d'une autre contrainte, appelée corrélation élevée, n'est pas assuré par la construction, il faut le prouver. Nous utilisons un résultat de [BKT20] qui résout le premier problème, il ne reste donc plus qu'à prouver la corrélation élevée, qui est la partie la plus technique de Chapitre 4.

Nous ne pouvons pas simplement adapter la preuve d'un autre problème de test de propriété de [BKT20], car elle utilise une propriété d'erreur unilatérale de leur fonction, que notre fonction ne satisfait pas. Nous espérons que la manière dont nous surmontons ce problème aidera les recherches futures à prouver des bornes inférieures pour d'autres problèmes, voire à obtenir une méthode de borne inférieure plus générale et plus facile à utiliser que la méthode polynomiale duale.

Enfin, nous montrons qu'il existe des propriétés de graphes, à la fois dans le modèle à degré limité et dans le modèle dense, qui ont une complexité de requête quantique essentiellement maximale. Ces deux contributions sont des adaptations de résultats similaires dans le cadre classique, et elles reposent sur la difficulté de distinguer les séquences uniformes des séquences « indépendantes k à k ».

Chapter 1

Introduction

1.1 Context

Nowadays, the amount of structured digital data in our lives grows at an exceeding rate which makes efficient information processing very important. For example, the ensemble of all the scientific articles can be represented as a gigantic, directed network of nodes (articles) connected by edges (references between the articles), and one could be interested in finding an influential article, i.e. one with more than a certain number of citations. Other examples of large networks include the world wide web, social networks, all the messages in a messaging application, etc. Graphs are of paramount importance, when it comes to understanding large networks like these, since they provide a natural way to represent and analyse complex relationships inside datasets.

In general, having access to an input object of huge size, an algorithm has to solve a problem, for example to decide whether it satisfies some property. Sometimes even merely reading the whole data would require too much time, in this case we would like to have a *sublinear algorithm* that solves the problem. Several different paradigms aim at achieving this. One possible way is to decrease the amount of data via random sampling: by making some assumption about the input and only looking at a small part of it, in some cases the problem can still be solved with high probability. Another possibility is to use quantum phenomena that can potentially speed up the computation.

On a high level, in this thesis we examine the computational speedup these techniques can provide. On the one hand this means designing new, efficient algorithms that solve a given problem; and on the other hand, proving lower bounds showing that no algorithm that solves the problem can be more efficient than the lower bound. Moreover, one can compare how efficiently a task can be solved in different models. For example, it is a major question in quantum computing to find useful problems with an exponential quantum advantage, meaning that a quantum computer can solve them much more efficiently than a classical machine. Hence, improving the classical efficiency of a problem like this by giving a new algorithm is also interesting from the quantum computing perspective.

In the following, let us look at the basic models and historical background of some relevant fields.

1.1.1 Classical computing and query complexity

Classical computing

The theory of computing dates back to the 1930s and the work of Kurt Gödel, Alonzo Church, Alan Turing and others. In particular, Turing introduced one of the most common computational models, that we now call the *Turing machine* [Tur37].

Turing machines are important because in a way they formalise what we can call a computational process or an *algorithm*. Moreover, according to the extended Church-Turing thesis, *efficient computations* correspond to efficient Turing machines. By this we mean that the amount of time (number of elementary steps) and space (memory) necessary for executing the computation are approximately the same in different models. From now on we will talk about algorithms.

A *deterministic* algorithm's actions at any step are determined by its internal state and the input. In addition to this, a *probabilistic* (or randomised) algorithm also has access to some randomness which can be thought of as a coin that it can toss. The machine's actions can also depend on the outcome of this randomness, and in this model, it usually suffices to succeed with probability 2/3. A third model is *nondeterministic* algorithms that can make different actions in parallel, and it suffices if any of these computations succeed.

In complexity theory, an algorithm usually solves a *decision problem*, meaning that it receives an input from a universe $x \in U$, and it has to output a bit representing whether x satisfies some property, i.e. whether $x \in \mathcal{P}$ for some subset $\mathcal{P} \subseteq U$. Function problems, where the task is to output something more complex than a bit, can often be reduced to decision problems with little overhead in the complexity. The time (resp. space) complexity of a decision problem on input x is the number of elementary steps (bits of memory) the best possible algorithm makes (uses) in order to decide whether the input satisfies the property. In the following we focus on time complexity.

The time complexity of a problem usually highly depends on the length of the input, for example deciding if the word "quantum" appears in the title of this thesis takes much less time than searching for it in the Bible. This is why time complexity is usually expressed as a function of the input length, where the length is usually denoted by n (or N), and the time complexity by $T(n)$. This way, the time complexity $T(n)$ of a decision problem on inputs of length n is the number of elementary steps the algorithm needs to make in order to decide for any input of length n whether it satisfies the property.

Moreover, constant factors are usually omitted, and only the order or growth rate of $T(n)$ is considered, which results in using the asymptotic notations $O, \Omega, \Theta, o, \omega$. For example, if $T(n) \leq 4n^2+2n-5$ then we say $T(n) \in O(n^2)$; and if $T(n) \geq 2\sqrt{n}-n^{1/3}-1$ then we say $T(n) \in \Omega(\sqrt{n})$. In this model, a computation is usually considered to be efficient if its time complexity is at most some polynomial function of the input length.

Query complexity

Query complexity is a model of computation where the input is not given to the algorithm explicitly, but as a "black box", and the algorithm has to make queries in order to gain information about it. This is also called oracle access: in each query the algorithm asks "What is the i-th character of the input?", and the oracle provides the corresponding character. The cost of the algorithm is the number of queries it makes for deciding whether the input satisfies a given property.

For example, let the input be a sequence of bits $x = 001010$ that the algorithm does not know, but it has access to the query operator $\mathcal{O}_x$. When querying index 2, the query operator returns the second bit of x: $\mathcal{O}_x(2) = x_2 = 0$. An example of a property is the OR function, i.e. the set of inputs having at least a 1-bit. In this case, query complexity measures the number of queries necessary (lower bound) or sufficient (upper bound) to decide if any input sequence has at least a 1-bit.

A query algorithm corresponds to a decision tree: the first query it makes is the root of the tree and the possible outcomes of the query correspond to the edges incident to this node. Based on the outcome, and on some randomness in the case of a probabilistic algorithm, another query can be made, which corresponds to the node at the other end of the edge of the first query, and so on.

Similarly to time complexity, query complexity is usually parameterised by the input length n: it is the number of queries necessary to solve the problem on any input of length n. Query complexity can be associated with sublinear algorithms because making n queries to an input of length n is trivially sufficient: in n queries the algorithm can learn the whole input and it does not need to make any further queries. Thus, query complexity is interesting when it is sublinear, i.e. $o(n)$.

Query complexity provides a natural model of several scenarios, for example when the input is stored on the server of a company. More generally, if the algorithm has restricted access to a large, distant object, then it would take a lot of time or money to obtain the whole of it. Then it is a reasonable goal to minimise the number of queries, or the amount of information needed for the algorithm.

Compared to time complexity, this model is useful because it makes it possible to prove *lower bounds* on the complexity of many problems. A lower bound $f(n)$ on the query complexity of a problem means showing that for any algorithm there is an input of length n such that the algorithm needs to make at least $f(n)$ queries to solve the problem on this input. Lower bounds are important because they make it possible to know the exact query complexity of a problem, otherwise we would only know upper bounds on them. This way it is possible to show that an algorithm for a problem is essentially query optimal, i.e. there cannot exist a more efficient way of solving it.

For example, it is in the query complexity model that one can show a lower bound on the problem of sorting, using a comparison oracle. In particular, we have n elements, and the way we can access them is to query pairs of elements and learn their relative order. In this model, one can show that sorting the n elements takes $\Omega(n \log n)$ queries (while there are $\binom{n}{2}$ element pairs). The well-known merge sort algorithm runs using $O(n \log n)$ comparisons, so it is essentially optimal.

1.1.2 Property testing

Instead of total decision problems, where the algorithm has to decide whether the input satisfies a property or not, one can consider partial decision problems or *promise problems*, where the algorithm can assume that the inputs it receives satisfy some promise. This way, the task becomes easier: instead of partitioning the whole input space to yes and no instances, it has to do so only for a subset of the input space. This means that there is no guarantee on the algorithm's output for inputs that do not satisfy the promise. Promise problems were introduced in [ESY84].

An important setting where using query complexity is natural, is *property testing*. It focuses on designing ultrafast algorithms (also known as testers) that read only a small part of the input and distinguish inputs that satisfy some property from inputs that are "ε-far" from satisfying it for some distance measure and parameter $\varepsilon \in (0, 1)$, that is usually considered to be constant. Notice that this is a promise problem: the inputs are promised to either satisfy the property or be far from it. This field was initiated in the work of [BK89] where the authors designed testers for checking the output of programs. Later, several works considered this verification or self-testing aspect of property testing, see [BLR90, EKK$^+$98] and the references therein.

As a possible use-case, when the exact computation is expensive, one can use property testing algorithms as a precursor to running the final algorithm. If the input does not pass the property testing test, we can safely reject it, without running the expensive final computation. Alternatively, property testing algorithms can be seen as approximation algorithms for decision problems: instead of outputting an approximately correct value, it makes a decision that is correct at least for some object that is close to the input. A third motivation is that if a property testing algorithm usually makes the right decision, then it is a heuristic with an additional provable guarantee, because we know that for some inputs it is going to succeed with high probability.

Goldreich, Goldwasser, and Ron [GGR98] were the first to consider graphs in the context of property testing. Formally, given some form of query access to an unknown graph G on N vertices, and a property $\mathcal{P}$ of interest, the goal is to distinguish with high probability if G satisfies the property $\mathcal{P}$, or whether it is far from all graphs that satisfy $\mathcal{P}$, with a suitable notion of farness. In [GGR98] the "dense" graph model was considered, where a graph is accessed through *adjacency matrix queries*: querying a pair of vertices (u, v) reveals whether u and v are linked by an edge in the graph. In this model, a graph G is ε-far from satisfying $\mathcal{P}$ if one needs to add or remove at least εN^2 edges of G to obtain a graph that satisfies $\mathcal{P}$.

Interestingly, there are some problems that are notoriously hard if we want to solve them exactly, but their property testing version in the dense graph model is surprisingly easy. For example, deciding if a graph contains a Hamiltonian cycle is not expected to be solvable in polynomial time. But the property testing version is trivial, since from any N-vertex graph, by adding at most N edges, we can obtain a graph with a Hamiltonian cycle, thus in the dense graph model any graph is $1/N$-close to being Hamiltonian. Another example is 3-colourability, that is also difficult to solve exactly, but for property testing in the dense graph model, it suffices to sample $O(1/\varepsilon^3)$ vertices and check if the corresponding induced subgraph is 3-colourable [GGR98].

In a later work, Goldreich and Ron [GR02] introduced the "bounded-degree" model for testing sparse graphs, focusing on the properties of bipartiteness and expansion. In this model, a d-bounded degree graph G with N vertices is accessed by performing *adjacency list queries*: for a vertex v and an integer $i \in [d]$, the query (v, i) returns either the i-th neighbour of v, or some special symbol if v has less than i neighbours. The graph G is said to be ε-far from some property $\mathcal{P}$, if one needs to add or delete at least εdN edges of G to obtain a graph that satisfies $\mathcal{P}$. Over the last two decades, there has been a significant number of works in this model, and we refer the interested reader to the books by Goldreich [Gol17] and Bhattacharyya and Yoshida [BY22] and several surveys [Fis01, Ron10, CS10, RS11].

There is a version of property testing, where instead of distinguishing inputs that satisfy the property from those that are ε-far, the task is to distinguish inputs that are ε'-close vs ε-far from the property, where $\varepsilon' < \varepsilon$. This is called *tolerant property testing* and it is closely related to distance approximation [PRR06]. It is clearly a generalisation of usual property testing since setting $\varepsilon' = 0$ yields the usual, non-tolerant version.

1.1.3 Quantum computing

The history of quantum computing began in the 1980s with the works of Yuri Manin, Paul Benioff and Richard Feynman [Man80, Ben80, Fey82]. The theoretical model is the quantum version of the Turing machine introduced by Benioff [Ben80]. The idea was further developed by David Deutsch [Deu85], who also suggested another model, based on quantum gates, similarly to classical logic gates in circuits.

Quantum computers use quantum phenomena to perform computation in superposition. This makes the model somewhat similar to the classical models of probabilistic (randomised) and nondeterministic computing. Quantum computing is more powerful than probabilistic computing, meaning that for every randomised algorithm there exists a quantum algorithm that solves the same task in the same complexity. Quantum and nondeterministic computing are incomparable.

The field of quantum computing has significantly influenced many computer science paradigms, including cryptography, algorithms, and large-scale data processing. This new perspective on computer science, based on quantum physics, has sparked many fresh research directions. This includes the topic of this thesis, which combines quantum computing, property testing and algebraic topology.

One of the fascinating aspects of quantum computing is that is permits to have sublinear algorithms in cases where classically it is not possible. For example, a classical algorithm looking for a “marked” element is a database of size n needs $\Omega(n)$ time to find it. For a quantum computer, solving this task only takes $O(\sqrt{n})$ time, using the famous Grover search algorithm [Gro96].

Let us recall that in classical query complexity, the algorithm has access to the input via the query operator $\mathcal{O}_x$. This operator can be modified to make it fit for dealing with quantum algorithms that can make queries in superposition. In this case, we obtain the notion of *quantum query complexity*. Similarly to its classical counterpart, this model is useful because it makes it possible to prove *lower bounds* on the complexity of many problems.

Lower bounds are important if we want to prove separations between different models. For example, Shor’s celebrated quantum algorithm solves integer factorisation in polynomial time [Sho94], but no polynomial time classical algorithm is known for this problem. However, no classical lower bound is known either, that would prove that any classical algorithm needs more than polynomial time. Thus, for this extremely important problem, the superpolynomial separation between classical and quantum algorithms is not proved, it is only according to the current state of the art.

But there are other problems where similar separations between classical and quantum algorithms are proved. One of the first examples of an exponential quantum advantage is the Deutsch-Jozsa algorithm [DJ92], where we are given query access to a Boolean function $f : \{0,1\}^n \to \{0,1\}$ with the promise that it is either constant (takes the same value for any input) or balanced (takes value 0 on half of the domain and 1 on the other half). The algorithm has to decide exactly (with error probability 0) which one is the case. Classically this clearly takes $2^{n-1}+1$ queries, but the Deutsch-Jozsa quantum algorithm solves the task with a single query.

A similar result exists in the bounded-error case. Given query access to a function $f:\{0,1\}^n \to \{0,1\}$ that is promised to satisfy $f(x)=x\cdot s$ for a hidden $s\in\{0,1\}^n$, find s. The Bernstein-Vazirani quantum algorithm solves this task exactly with a single query [BV97]. However, it can be shown that any bounded-error randomised algorithm needs to make $\Omega(n)$ queries.

1.1.4 Topological data analysis

Recently, there has been an increasing interest in using simplicial complexes to model higher-order relations in data sets – a technique often called *topological data analysis* (TDA) [Car09]. One of the reasons for this is that it seems to be useful: it has found applications in several domains, like in machine learning or in the analysis of images and networks, that can be used for example in oncology and cosmology [BAD21, PEv$^+$16]. Another reason is that it appears to be a candidate for a natural, useful task that could admit an exponential quantum advantage in some cases.

Simplicial complexes are set families that are closed under the subset relation. They can alternatively be viewed as hypergraphs with the additional constraint that if a hyperedge is in the hypergraph then all its subsets are there too. A special case of simplicial complexes is *clique complexes*, that are defined by a graph: each clique of the graph is a set in the complex. A set of size $(k+1)$ that is in the complex, is called a k-face or a k*-simplex*.

In order to robustly classify data, a feature of particular importance is the rank of the homology groups, called the *Betti numbers* of the simplicial complex, which intuitively characterise the number of high-dimensional holes. This theory is called simplicial homology, which belongs to the broader branch of algebraic topology. In particular, the so-called *persistent Betti numbers* have been useful for applications because they capture a scale-independent global property of the data set [PEv$^+$16, KMH$^+$21, BAD21].

In a recent paper, the authors showed that it is QMA1-hard to compute Betti numbers, or to estimate them with relative error, even in the special case of clique complexes [CK24]. This is why the focus has been on the additive approximation of the (normalised) Betti numbers. In [LGZ16] the authors proposed a quantum algorithm that solves this problem in polynomial time for a set of parameters: the runtime is poly$(n, 1/\gamma, 1/\varepsilon)$ where n is the number of vertices, ε is the additive precision and γ is the (normalised) spectral gap of the so-called combinatorial Laplacian of the complex. So far, most applications need the exact computation of low dimensional Betti numbers, hence the estimation of high dimensional ones was not in the focus. This is why the LGZ algorithm was somewhat unfairly compared to classical algorithms for Betti number computation (rather than estimation), which take time exponential in the dimension k. This suggested that there could be an exponential quantum advantage for this natural, useful task.

1.1.5 Collision finding and related problems

Collision finding is a ubiquitous problem in the field of algorithm theory with many applications in cryptography, algorithms, statistics etc. Here we are given a list of bounded integers $s = (s_1, \dots, s_N) \in [R]^N$ and the task is to *find* a pair $i \neq j$ such that $s_i = s_j$. For example, collision resistance is an important property of cryptographic hash functions, that have numerous information-security applications. Collision resistance means that on a "good" hash function it is difficult to solve the collision finding problem (if we represent the hash function as a list of integers).

This original version of the problem is well-understood both classically and quantumly: the classical query complexity of the collision problem is $\Theta(\sqrt{n})$ by the birthday paradox; and its quantum query complexity is $\Theta(n^{1/3})$ by the BHT algorithm [BHT98] and the lower bound of Aaronson and Shi [AS04].

However, there are some modified or generalised versions that need further research. For example, we can consider k-collisions, where in the list of integers $s = (s_1, \dots, s_N) \in [R]^N$ we want to find a k-tuple $i_1, \dots, i_k$ such that $s_{i_1} = s_{i_k}$. Classically, it was shown that the query complexity of this problem is $\Theta(n^{1-1/k})$ [HS12, PW23]. The quantum query complexity is only settled in the special case where the input is a random string of integers, and then the complexity is $\Theta\left(n^{\frac{1}{2}\left(1-\frac{1}{2^k-1}\right)}\right)$ [LZ19].

Instead of finding, we can consider a decision version of this problem, where the algorithm has to *distinguish* with high probability inputs that contain at least εN collision pairs from those that do not contain any. This version could potentially be simpler to solve, but the classical lower bound [PW23] and the quantum lower bound for the $k = 2$ case [AS04] were actually proved for the decision version. One of our goals in this thesis is to give a lower bound for this version in the quantum setting for general k.

This version of the problem can be further generalised to graphs and it can be considered in the context of property testing. In fact, in [HS12] the authors gave an algorithm for this more general, subgraph-freeness testing problem. For proving their lower bound on the decision version of the k-collision problem, [PW23] used the proportional moments technique of [RRSS09]. Then they gave reductions to obtain lower bounds on the general subgraph-freeness property testing problem in bounded-degree directed graphs. This successful lower bound technique of [RRSS09] has no quantum analogue yet, and a future goal of ours is to extend it to this setting.

1.2 Overview of the results

Let us provide a short, high-level overview of the main contributions of this thesis.

1.2.1 Classical algorithms for Betti number estimation

Additive approximation of Betti numbers

In Chapter 3, the results of two articles are presented. First, based on **[AGSS23]**, we give a classical algorithm for the following problem. The input is a simplicial complex K with n vertices, d_k many k-simplices and k-th Betti number β_k. The output is an ε-additive estimate of the normalised Betti number β_k/d_k. We assume we have sampling and query access to the input simplicial complex K. We can alternatively say that in polynomial time we can check for a set of vertices if it is a simplex in K, and we can obtain a random simplex of a given size.

An important element of our algorithm is a matrix H related to the so-called combinatorial Laplacian of the complex K. We show that the normalised trace of a high enough power of H gives an estimate of the k-th normalised Betti number. Then we estimate this normalised trace by noticing that it corresponds to the expectation of a random variable that can be calculated by a Monte Carlo process.

Intuitively, we start from a random k-simplex of K and do a random walk over the k-simplices of K with transition probabilities corresponding to the entries of H. Using a standard concentration bound, we can upper bound the number of times we have to repeat this process in order to obtain a good enough approximation of the k-th normalised Betti number.

The complexity of this basic algorithm can be slightly improved by using Chebyshev polynomials to estimate the power of H. Moreover, for the special case where K is a clique complex, we can show that H is sparser than in general, which reduces the number of repetitions necessary for obtaining the desired estimation.

It was already known before our work that quantum algorithms can solve this task efficiently, even for large k, but no efficient classical algorithm was known in this regime. More precisely, the time complexity of the quantum algorithm of [LGZ16] is polynomial in n, $1/\varepsilon$ and $1/\gamma$, where n is the number of vertices, ε is the additive precision parameter and γ is the spectral gap of the combinatorial Laplacian.

This way, our algorithm serves as a classical benchmark of quantum ones, because it shows that the problem can be solved in polynomial time even classically, although for a more restricted set of parameters than in the quantum case. In particular, for general simplicial complexes our algorithm runs in polynomial time if ε and γ are constants. In the special case of clique complexes, we have a slightly improved result: for instance, if $k \in \Omega(n)$ and γ is constant, then we can allow ε to be inverse polynomial in n.

Property testing very large Betti numbers

In the same chapter, based on another article **[SA25]**, we examine simplicial homology in the setting of property testing. Having query access to the underlying graph of a clique complex, the algorithm has to distinguish if β_k is nearly the maximum possible, or the input is ε-far from this. We show that the query complexity of this task depends only on the proximity parameter ε, i.e. it is independent of the input size.

We use a notion of independence of k-simplices that comes from matroid theory, and we denote the maximum number of independent k-simplices as r_k. We prove that r_k cannot be much smaller than the total number of k-simplices d_k. Moreover, with this notion it is possible to have an elegant expression of β_k that includes d_k, r_k and r_{k-1}.

Then we prove our result, first in the special case of $k = 0$, and then in general. The proofs use the previously mentioned formulas to show that a very large k-th Betti number means few independent $(k+1)$-simplices, which also means few $(k+1)$-simplices in total. In a clique complex a $(k+1)$-simplex is a $(k+2)$-clique, so in this case the graph is close to not containing any $(k+2)$-cliques. Also, we can show that being far from having a large β_k implies being far from not containing any $(k+2)$-cliques. This way, we reduce our problem to the tolerant property testing of $(k+2)$-clique-freeness. This is known to have query complexity that only depends on the proximity parameters, using well-known results [Für95, FN07].

1.2.2 Quantum property testing

Our work presented in Chapter 4 is based on **[AMSS25]**, and it initiates the study of quantum property testing of directed graphs in the bounded degree model, where the algorithm has query access to the adjacency list of the outgoing edges of the graph's vertices. We consider a subgraph-freeness testing problem that is a generalisation of k-star-freeness for constant k. A k-star is a graph with $(k+1)$ vertices: one centre vertex and k outer vertices, and there is an edge from each outer vertex to the centre.

We recall the example at the beginning of this introduction, where scientific articles refer to each other. This fits into this model: when reading an article, we only see the articles it cites, but we want to find an article that is cited by many (k) others, which is exactly a k-star.

We connect the subgraph-freeness problem to the k-collision problem in multiple ways. First, we give an algorithm for our subgraph-freeness problem by generalising the k-collision finding algorithm of [LZ19] to graphs. Moreover, we need to prove that this approach works in the property testing context. In particular, they assume that their input contains many k-collisions, but we have to prove that an input that is far from subgraph-freeness, contains many subgraphs that are disjoint in some appropriate sense.

Intuitively, for the special case of k-stars, our algorithm first samples some vertices and queries their neighbours. Then Grover search is used to look for other vertices that have an already queried neighbour in their neighbourhood, and thus several 2-stars are found. With iterated Grover searches, some of these stars grow into larger and larger stars until one k-star is found.

For the lower bound, we give a simple reduction from the property testing of k-collision-freeness to the k-star-freeness problem. This way, it suffices to give a lower bound on the former problem, or even a special case of it. In particular, we consider the problem where one has to distinguish between no k-collisions and many distinct values where a k-collision occurs. This problem can be easily formulated as a composition of simple Boolean functions (a promise version of OR and the threshold function) if we use an appropriate binary encoding of the input.

The lower bound method we use is called the dual polynomial method, where one needs to provide a polynomial – that we call the dual polynomial – that certifies that any quantum algorithm needs to make many queries in order to solve the problem. For some basic functions there exist known dual polynomials, and there are also some ways to create a potential dual polynomial for a composition of these simple functions. This is why it is important to consider a version of our problem that we can formulate like this.

The difficulty is that the polynomial we obtain by using these building blocks, is not yet a good dual polynomial, because it does not satisfy one of the constraints it should. Moreover, satisfying another constraint, called high correlation, is not provided by the construction, one needs to prove it. We use a result from [BKT20] which fixes the first issue, so the only remaining task is to prove the high correlation, which is the most technical part of Chapter 4.

We cannot simply adapt the proof of another property testing problem from [BKT20], because it uses a one-sided error property of their function, that our function does not satisfy. We hope that the way we overcome this issue is going to help future research to prove lower bounds for other problems, or even to obtain a more general lower bound method that is easier to use than the dual polynomial method.

Finally, we show that there are graph properties, both in the bounded degree and in the dense model, that have essentially maximal quantum query complexity. Both these contributions are adaptations of similar results in the classical setting, and they rely on the hardness of distinguishing uniform sequences from "k-wise independent" ones.

1.3 Publications

This thesis is based on the following articles.

Chapter 3:

– **[AGSS23]**: Simon Apers, Sander Gribling, Sayantan Sen, and Dániel Szabó. A (simple) classical algorithm for estimating Betti numbers. Quantum, 7:1202, 2023. DOI: 10.22331/q-2023-12-06-1202.

– **[SA25]**: Dániel Szabó and Simon Apers. Holey graphs: very large Betti numbers are testable. In SOFSEM 2025: Theory and Practice of Computer Science, pages 298–310. Springer Nature Switzerland, 2025. DOI: 10.1007/978-3-031-82697-9_22.

Chapter 4:

– **[AMSS25]**: Simon Apers, Frédéric Magniez, Sayantan Sen, and Dániel Szabó. Quantum property testing in sparse directed graphs. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), volume 353, pages 32:1–32:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2025.32.

Chapter 2

Preliminaries

In this chapter we introduce some notations, models and results that we will use throughout the thesis. In later chapters we will remind the reader of some of these before using them.

2.1 Notations and basic definitions

Let us denote $[n] = \{1, \dots, n\}$ and $[n]_0 = \{0, \dots, n\}$. Let $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ respectively denote the set of nonnegative integers, integers, rational numbers, real numbers and complex numbers. Moreover, for any prime number p let $\mathbb{F}_p$ denote the finite field of integers modulo p.

For a set S, its size (cardinality) or the number of its elements is denoted by $|S|$. The power set of S (i.e. the set family of all subsets of S) is denoted by 2^S. For any nonnegative integer $k \leq |S|$, the set family of all the size-k subsets of S is denoted by $\binom{S}{k}$. Note that the size of set family $\binom{S}{k}$ is exactly the usual binomial coefficient $\binom{|S|}{k}$, and the size of 2^S is $2^{|S|}$.

If we allow repetitions (but the order does still not count), we get multisets, and the set of size-k multisets with elements from S is denoted by $\left(\!\binom{S}{k}\!\right)$. The size of this set is $\left(\!\!\binom{|S|}{k}\!\!\right) = \binom{|S|+k-1}{k}$, also known as the multiset coefficient or the number of k-element combinations of $|S|$ objects with repetition.

A string (of characters) is some $x \in \Sigma^n$ where Σ is the alphabet and n is the length of x. In the special case of bitstrings, $\Sigma = \{0, 1\}$. The Hamming weight of a bitstring $x \in \{0, 1\}^n$ is $|x|_H = |\{i \in [n] : x_i = 1\}|$, i.e. it is the number of 1s in x.

When writing $\log$ without specifying the base, we mean $\log_2$ (binary logarithm). Natural logarithm is denoted by $\ln$. On the other hand, when using notation $\exp(f)$, we mean e^f. The expected value of a random variable X is $\mathbb{E}[X]$, and more generally its kth moment is $\mathbb{E}[X^k]$. The sign function sgn assigns -1 to negative values, 1 to nonnegative values. When it is applied on vectors it acts elementwise.

The usual composition of two functions $f : A \to B$ and $g : B \to C$, is denoted by $g \circ f : A \to C$, and it is the application of g on the result of f, i.e. for any $x \in A$, $(g \circ f)(x) = g(f(x))$. In the case of multivariate functions $f : A^n \to B$ and $g : B^m \to C$, we define a different kind of composition: $g \odot f : A^{nm} \to C$, such that for any $x \in A^{nm}$ also denoted as $x = (x_1, x_2, \dots, x_m)$ with $\forall i \in [m]\ x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n}) \in A^n$, $(g \odot f)(x) = g(f(x_1), f(x_2), \dots, f(x_m))$.

Notations $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$ hide constant factors and dependencies on parameters that are considered to be constants. Moreover, when augmented with the tilde notation they also hide factors that are polylogarithmic in the argument. For example, $f(n) \in \tilde{O}(g(n))$ means that there exists some constant k such that $f(n) \in O(g(n) \log^k g(n))$.

Graph theory

An *undirected graph* $G = (V, E)$ is a pair of a vertex set V and an edge set E. The latter consists of edges that are unordered pairs of vertices: for $u, v \in V$ saying that u and v are connected by an edge is equivalent to $\{u, v\} \in E$. We say that there is a *path* between $s = v_0$ and $t = v_{l+1}$ (with $s, t \in V$) if there exists an integer l and vertices $v_1, \ldots, v_l \in V$ such that $v_0, v_1, \ldots, v_{l+1}$ are all distinct and $\forall i \in [l]_0 : \{v_i, v_{i+1}\} \in E$. A graph $G = (V, E)$ is called *connected* if for every $u \in V$ and $v \in V \setminus \{u\}$, there exists a path between u and v.

A *digraph* or directed graph is like an undirected one, but the edges are directed (and are often called arcs): a directed edge is an ordered pair of vertices, and $(u, v) \in E$ means that there is a directed edge from u to v. Similarly to the undirected case, we say that there is a *directed path* from $s = v_0$ to $t = v_{l+1}$ (with $s, t \in V$) if there exists an integer l and vertices $v_1, \ldots, v_l \in V$ such that $v_0, v_1, \ldots, v_{l+1}$ are all distinct and $\forall i \in [l]_0 : (v_i, v_{i+1}) \in E$. A digraph $G = (V, E)$ is called *strongly connected* if for every $u \in V$ and $v \in V \setminus \{u\}$, there exists a directed path from u to v (and thus from v to u as well).

In a context where there are several graphs, we can emphasise which graph's vertices or edges we mean: for a graph G its vertex set is $V(G)$ and its edge set is $E(G)$. A *subgraph* of an undirected (resp. directed) graph $G = (V, E)$ is any graph $G' = (V', E')$ satisfying $V' \subseteq V$, $E' \subseteq E$ and $E' \subseteq \left(\!\binom{V'}{2}\!\right)$ (resp. $E' \subseteq V' \times V'$). Notice that these definitions allow self-loops (i.e. edge from a vertex v to itself) but no parallel edges of the same kind (i.e. no two edges from vertex v to u).

2.2 Query complexity

In the query complexity model, we consider inputs $x \in \Sigma^I$ over a finite alphabet Σ and indexed by a set I. They are not given explicitly to the algorithm. Instead, the algorithm has *query access* (or black box access) to an input oracle $\mathcal{O}_x$ encoding x. This means that the algorithm can query each character of x: for any $i \in I$ it learns $\mathcal{O}_x(i) = x_i$.

As examples, let us look at two kinds of inputs that are ubiquitous in discrete mathematics: strings of integers and graphs. When the input is a string $s = (s_1, \ldots, s_N)$ of positive integers $\leq R$, then $I = [N]$ and $\Sigma = [R]$.

In the case of graphs, there are multiple options. In the *dense graph model*, the algorithm has query access to the adjacency matrix of the graph. For a graph with vertex set V, this yields $I = V \times V$ and $\Sigma = \{0, 1\}$. That is, querying a vertex pair (u, v) tells us if they are connected by an edge in the graph or not. For undirected graphs, querying (u, v) and (v, u) is equivalent, but in the directed case the order matters.

In the *bounded degree model*, we query the adjacency list of the graph. For a graph with vertex set V and degree bound d, we have $I = V \times [d]$ and $\Sigma = V \cup \{\perp\}$. Here we assume that each vertex fixed an order of its neighbours, and a query (v, i) returns the i-th neighbour of v if it exists, and a special character $\perp$ otherwise. In the case of undirected graphs, this is well-defined, but for directed graphs there are two options. In the *unidirectional model*, the algorithm only has access to the list of out-neighbours. On the other hand, in the *bidirectional model*, there are two lists: one for the out-neighbours and one for the in-neighbours, so the algorithm has to specify if it would like to learn the i-th in- or out-neighbour of v. This can be achieved, for example, by the query containing one further bit.

A *property* is a predicate $P : D_P \to \{0, 1\}$ over domain $D_P \subseteq \Sigma^I$. Equivalently, it can be interpreted as a subset of the domain: $\{x \in D_P : P(x) = 1\}$ (in Section 2.3 this subset is denoted by $\mathcal{P} = \Pi_{\text{YES}}$). The (randomised) *query complexity* of a property measures the minimum number of queries that an algorithm has to make in order to decide with high probability whether the property holds for any input. This is called *worst-case complexity* because the algorithm has to make the right decision for every input, even the "hardest" inputs (with high probability). This is in contrast with average case complexity, where the input is taken from a distribution and thus the error probability over both the algorithm's internal randomness and the input has to be small, i.e. on inputs that the algorithm receives with low probability, it can err. In this thesis, we only consider worst-case complexity.

2.3 Property testing

We assume that the input of an algorithm is taken from a universe that is some set U. A property $\mathcal{P}$ is a subset of U, and we say that an input $x \in U$ satisfies $\mathcal{P}$ if $x \in \mathcal{P}$. In *total decision problems*, the algorithm receives an input $x \in U$, and it has to decide if x satisfies a property $\mathcal{P}$ or not, i.e. if $x \in \mathcal{P}$ or $x \in U \setminus \mathcal{P}$.

In *promise problems* (or partial decision problems), the algorithm can assume that it will only receive inputs that satisfy a promise which is a subset of the universe $U' \subseteq U$. This way, the task becomes easier: the algorithm only has to distinguish between $x \in \Pi_{\text{YES}} = \mathcal{P} \cap U'$ or $x \in \Pi_{\text{NO}} = (U \setminus \mathcal{P}) \cap U'$. It is important to note that the algorithm may receive any input from U, but it only has to satisfy some guarantees for inputs from U'.

Gap problems are a special case of promise problems where the algorithm is promised to only receive inputs that are either in $\mathcal{P}$ or "ε-far" (according to some distance measure and parameter ε) from any input in $\mathcal{P}$. Here $\Pi_{\text{YES}} = \mathcal{P}$ and Π_{NO} is the set of inputs that are ε-far from $\mathcal{P}$.

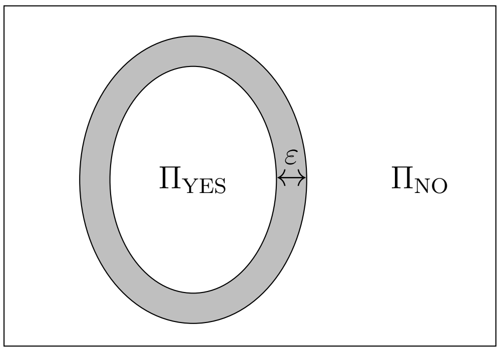

Figure 2.1 – Depiction of gap problems. The whole rectangle is U and the white area is U'.

In the case of *property testing*, the algorithm has to solve a gap problem with high success probability, i.e. it has to distinguish inputs that satisfy the property from those that are "far" from the property. Before defining property testing algorithms, we need to discuss the distance measure that is used in the definition.

The choice of distance measure usually depends on the query model considered. As discussed before, in general, query access can be viewed as black box access to the input $x \in \Sigma^I$ where querying an index $i \in I$ reveals $x_i \in \Sigma$. This way, the distance of two objects is defined as the proportion of positions where they differ. Again, a property $\mathcal{P}$ is just a set of inputs: $\mathcal{P} \subseteq \Sigma^I$.

Definition 2.3.1. *The distance of two inputs $x, y \in \Sigma^I$ is defined as*

$$\operatorname{dist}(x, y) = \frac{|\{i \in I : x_i \neq y_i\}|}{|I|},$$

We say that x is ε-far *from property $\mathcal{P}$ if $\operatorname{dist}(x, y) > \varepsilon$ for all $y \in \mathcal{P}$.*

In the case of graphs, the usual distance measures count the number of edges that one needs to add or delete in order to transform a graph G to another one G'. Since the number of vertices cannot change in this way, we can only compare graphs that have the same number of vertices: $|V(G)| = |V(G')|$. But in general, the two vertex sets can be different (i.e. we do not know how to relate the vertices in $V(G)$ to the ones in $V(G')$), that is why invariance under any permutation (or relabelling) of vertices is required. In fact, a *graph property* is defined as a set of graphs that is closed under graph isomorphism [Gol17].

Remark 2.3.2. *This difference, that for general properties we do not require permutation invariance but for graph properties we do, may seem strange at first, but it only reflects how the different objects are usually used. In most applications, the labels of the graph vertices are not important, we are only interested in the overall structure of the graph (e.g. is there a Hamiltonian cycle). But in string problems, the positions and values are often crucial (e.g. pattern matching) which means no permutation invariance. However, for some special string problems it can make sense to consider a permutation invariant distance measure: e.g. collision-type properties (like k-collision-freeness in Chapter 4) admit an invariance over permutations of both the positions and of the alphabet.*

For example, in the dense graph model (where we have query access to the adjacency matrix of the graph) the distance of two graphs G and G' with $|V(G)| = |V(G')|$ is defined as $\operatorname{dist}(G, G') = \frac{\min_\pi \{G \triangle \pi(G')\}}{n^2}$, where π is any permutation of the vertices and $\triangle$ denotes the symmetric difference of the two edge sets. So, the distance is the number of edges where the two graphs differ (up to isomorphism) divided by an upper bound on number of edges (sometimes the denominator is set to $\binom{n}{2}$ instead of n^2).

In the case of bounded-degree graphs with degree bound d (where we have query access to the adjacency list), the distance of two graphs is similarly defined as the number of edges where they differ (up to isomorphism) divided by $|V|d$.

Now let us define property testing and its tolerant version.

Definition 2.3.3. *Let $0 < \varepsilon < 1$ be a constant and $\mathcal{P}$ a property. A randomised algorithm $\mathcal{A}$ is an* ε-tester *for the property $\mathcal{P}$ if*

1. *For all $x \in \mathcal{P}$: $\Pr[\mathcal{A}(x) = \text{accept}] \geq 2/3$;*

2. *For all x that are ε-far from $\mathcal{P}$:* $\Pr[\mathcal{A}(x) = \text{accept}] \leq 1/3$.

The probabilities are taken over the randomness of $\mathcal{A}$.

Notice that no restriction is given on the acceptance probability of the property testing algorithm for inputs that do not satisfy $\mathcal{P}$ but are ε-close to it (the grey zone in Figure 2.1).

Definition 2.3.4. *Let $0 < \varepsilon_1 < \varepsilon_2 < 1$ be constants and $\mathcal{P}$ a property. A randomised algorithm $\mathcal{A}$ is an* ε_1-tolerant ε_2-tester *for the property $\mathcal{P}$ if*

1. *For all x that are ε_1-close to $\mathcal{P}$:* $\Pr[\mathcal{A}(x) = \text{accept}] \geq 2/3$*;*
2. *For all x that are ε_2-far from $\mathcal{P}$:* $\Pr[\mathcal{A}(x) = \text{accept}] \leq 1/3$.

The probabilities are taken over the randomness of $\mathcal{A}$.

We call a property $\mathcal{P}$ *(tolerantly) testable* if there is a (tolerant) property testing algorithm such that the number of queries it makes is independent of the input length, it only depends on the distance parameter(s).

2.4 Hoeffding's inequality

In the field of randomised algorithms, it a very common technique to repeat a random procedure several times in order to obtain better guarantees on the outcome, such as a smaller variance. One of the most well-known results here is Hoeffding's inequality. Some of our proofs depend on this result, and we are going to use two variants of it.

Lemma 2.4.1 (Hoeffding's Inequality [Hoe63], see also [DP09])**.** *Let $X_1, \ldots, X_t$ be independent random variables such that $a \leq X_i \leq b$ and let $X_{\text{avg}} = \frac{1}{t}\sum_{i=1}^{t} X_i$. Then, for any $\delta > 0$,*

$$\Pr\left[|X_{\text{avg}} - \mathbb{E}[X_{\text{avg}}]| \geq \delta\right] \leq 2\exp\left(\frac{-2t\delta^2}{(b-a)^2}\right).$$

For one sided deviation, the same result holds without the factor 2 in front of the exponential term. In the special case where all the X_i are identically distributed Bernoulli random variables (i.e. $a = 0$ and $b = 1$) this yields the following statement.

Corollary 2.4.2. *Let $X_1, \ldots, X_t$ be independent Bernoulli random variables with the same expected value $\mathbb{E}[X_1]$, and let $X_{\text{sum}} = \sum_{i=1}^{t} X_i$ (thus $\mathbb{E}[X_{\text{sum}}] = t \cdot \mathbb{E}[X_1]$). Then, for any $\delta > 0$,*

$$\Pr\left[X_{\text{sum}} - t \cdot \mathbb{E}[X_1] \geq \delta\right] \leq \exp\left(-2\delta^2/t\right).$$

2.5 Quantum computing

In classical computing, the unit of information is a bit. A bit can be in either one of two states that are commonly denoted by 0 and 1. Quantum mechanical phenomena show us that elementary particles can be in a superposition of states. Using these phenomena, one can create quantum bits or qubits: a qubit can be in a superposition of basis states 0 and 1. For more formal definitions let us start with some linear algebra.

2.5.1 Some linear algebra

A Hilbert space is a complete vector space equipped with an inner product. In this work, we use finite dimensional Hilbert spaces over the field of complex numbers $\mathbb{C}$. The complex conjugate of a number $z \in \mathbb{C}$ is $\bar{z}$, and its absolute value is $|z| = \sqrt{z\bar{z}} \geq 0$.

Using the Dirac or bra-ket notation, a (column) vector is denoted by $|v\rangle$ and its conjugate transpose by $\langle v|$. The inner product of two vectors $|u\rangle, |v\rangle \in \mathbb{C}^n$ is $\langle u|v\rangle = \sum_{i\in[n]} \bar{u}_i v_i \in \mathbb{C}$, and their outer product is $|u\rangle\langle v| = M \in \mathbb{C}^{n\times n}$ with $M_{i,j} = u_i \bar{v}_j$ for all $i, j \in [n]$.

Let p be a positive integer, then the p-norm of a vector $|v\rangle \in \mathbb{C}^n$ is defined as $\||v\rangle\|_p = \left(\sum_{i\in[n]} |v_i|^p\right)^{1/p}$. For example, the 1-norm is just the sum of the absolute values of the vector elements. For a matrix $M \in \mathbb{C}^{n\times m}$, its 1-norm is the maximum 1-norm of its columns $\|M\|_1 = \max_{j\in[m]} \sum_{i\in[n]} |M_{i,j}|$.

The trace of a matrix $M \in \mathbb{C}^{n\times n}$ is $\mathrm{Tr}(M) = \sum_{i\in[n]} M_{i,i}$. It is also equal to the sum of the eigenvalues of M (with multiplicities).

The adjoint (conjugate transpose) of a matrix M is denoted by M^*. A matrix $M \in \mathbb{C}^{n\times n}$ is called Hermitian if $M = M^*$; and it is called unitary if $MM^* = I$ where I is the identity matrix of size $n \times n$. A Hermitian matrix $M \in \mathbb{C}^{n\times n}$ is called positive semidefinite (or PSD) if for all $|v\rangle \in \mathbb{C}^n$, the real number $\langle v| M |v\rangle \geq 0$.

The usual tensor product of vector spaces $\mathbb{C}^a$ and $\mathbb{C}^b$ is denoted by $\mathbb{C}^a \otimes \mathbb{C}^b = \mathbb{C}^{ab}$. The tensor product of vectors $|u\rangle$ and $|v\rangle$ is $|u\rangle \otimes |v\rangle$ also denoted as $|u\rangle |v\rangle$, as $|u, v\rangle$ or even as $|uv\rangle$. The tensor product of matrices $M \in \mathbb{C}^{a_1\times b_1}$ and $M' \in \mathbb{C}^{a_2\times b_2}$ is $M \otimes M' \in \mathbb{C}^{a_1a_2\times b_1b_2}$.

2.5.2 The postulates of quantum computing

In this section we describe what are often called the four postulates of quantum computing.

Qubit A *qubit* is a unit vector from $\mathbb{C}^2$. We denote the standard basis vectors of this space by $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$. This way, any qubit can be expressed as $\alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. The scalars α and β are called *probability amplitudes*.

As α and β are complex numbers, they can alternatively be written in the exponential form: $\alpha|0\rangle + \beta|1\rangle = |\alpha|e^{i\varphi_\alpha}|0\rangle + |\beta|e^{i\varphi_\beta}|1\rangle = e^{i\varphi_\alpha}(|\alpha||0\rangle + |\beta|e^{i(\varphi_\beta-\varphi_\alpha)}|1\rangle)$. Here $|\alpha|$ and $|\beta|$ are reals from interval $[0,1]$ (still satisfying $|\alpha|^2 + |\beta|^2 = 1$); φ_α is called the global phase and the phase difference $\varphi = \varphi_\beta - \varphi_\alpha$ is called the relative phase. Both phases are reals from $[0, 2\pi)$. Changing the global phase has no observable consequences, it leaves the qubit unchanged. The relative phase φ is important, for example it explains interference, and thus it is called the *phase* of a qubit.

Register A system that consists of several qubits is called a *quantum register* and it is a vector in the tensor product space. For example, a 2-qubit system is a unit vector from $\mathbb{C}^4 = \mathbb{C}^2 \otimes \mathbb{C}^2$ that can be written as a linear combination of the standard basis vectors $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. Integers are often used instead of their binary encodings:

$|0\rangle, |1\rangle, |2\rangle, |3\rangle$. We will call unit vectors in Hilbert spaces *(quantum) states*[1].

If a quantum state can be written as a tensor product of individual qubits, it is called *separable*, and otherwise it is *entangled*. For example, $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is separable and $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is entangled. Entanglement is a commonly used phenomenon for example in quantum computing and communication.

Time evolution Quantum states can be transformed using unitary transformations, often called *quantum gates*. For example, if an n-qubit state $|\psi\rangle \in \mathbb{C}^{2^n}$ is transformed to another state $|\phi\rangle \in \mathbb{C}^{2^n}$ by unitary $U \in \mathbb{C}^{2^n \times 2^n}$, then we can write $U|\psi\rangle = |\phi\rangle$. Unitarity corresponds to reversible transformations.

Examples of commonly used, elementary quantum gates include the Pauli X (bit flip) and Z (phase flip) gates, the Hadamard gate H and the 2-qubit controlled-X (CNOT) gate. The T gate shifts the phase by $\pi/4$ and is linked to the Z gate by $Z = T^4$.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad H = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}, \quad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Measurement In this work we are not going to use precise details about measurements, so instead of a long formal definition it suffices to give a simple intuitive example. When a register is measured in the standard basis, it is "forced" into one of the standard basis states. If the n-qubit register being measured this way is $\sum_{j=0}^{2^n-1} \alpha_j |j\rangle$ then by the Born rule, the outcome of the measurement is $|j\rangle$ with probability $|\alpha_j|^2$ for all $j \in [2^n - 1]_0$.

2.5.3 Some models in quantum computing

There are several models of quantum computing (e.g. quantum Turing machine, adiabatic model, quantum annealing), in this thesis only the *circuit model* is considered. Here a procedure can be given as a quantum circuit, similarly to classical logic circuits.

Circuit model

In a quantum circuit, qubits (or registers) are drawn as horizontal lines with their initial value written on the left-hand side. Then a sequence of quantum gates $\boxed{U}$ (for unitary U) and measurements $\boxed{\nearrow}$ is applied on them (from left to right). The contol of controlled operations is denoted by $\bullet$ and it means that the application of the linked gate depends on the value on this wire. After a measurement, the value on the wire becomes classical which is denoted by a double wire.

It is preferable that a quantum circuit only contains small, 1 or 2-qubit *elementary gates* that are relatively easy to implement physically. Similarly to their classical counterparts, quantum circuits also have *universal gate sets* that are sufficient to implement any unitary

1. In fact, these vectors are usually called pure states. More generally, a quantum state can be a mixed state that is a probabilistic combination of pure states and is commonly represented by a density matrix. In this work, we are not going to use mixed states.

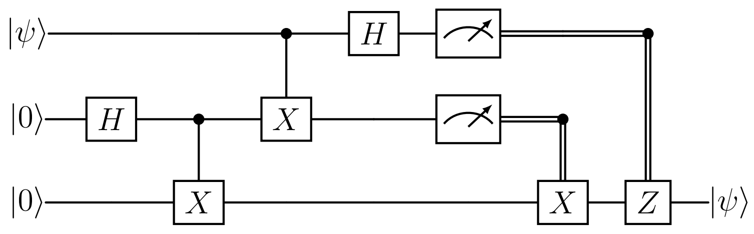

Figure 2.2 – Example of a circuit that implements quantum teleportation.

transformation with high precision without making the circuit too large. An example of a universal gate set is $\{H, T, CNOT\}$. Taking accuracy into account makes this question more subtle, but in this thesis we do not go into more details.

An example of a quantum circuit can be seen in Figure 2.2. It serves as a depiction of the mentioned elements, and we do not go into details about each step of it. The main idea is that using entanglement and classical communication it is possible to "teleport" any qubit $|\psi\rangle$. By this we mean that originally, we have $|\psi\rangle$ somewhere, but at the end of the protocol it can be far away from it, without moving the original qubit.

According to the *deferred measurements principle* (see e.g. [AKN98]), every intermediate measurement can be postponed to the end of the computation without changing the output distribution. The transformation that allows this, introduces as many extra qubits as the number of postponed measurements. This way, a circuit can be represented by a single unitary transformation followed by some measurements. This is often useful, for example when using some lower bound methods (see e.g. Section 2.5.4).

If our quantum algorithm solves a decision problem, the output depends on some measurement outcomes. As there is inherent randomness involved in quantum algorithms, we are usually satisfied with bounded error. Hence, we say that the algorithm solves the problem if it has small error probability. The space complexity of a quantum algorithm is the number of qubits it uses, and its time complexity is the number of elementary gates in the circuit.

Quantum query complexity

Quantum query complexity is similar to the classical version discussed earlier, but now the query operator has to be unitary, and in the case of input $x \in \Sigma^I$ one way to define it is as follows: $\mathcal{O}_x |i\rangle |z\rangle = |i\rangle |z \oplus x_i\rangle$, for $z \in \Sigma$ and $i \in I$, where $\oplus$ is usually the sum modulo $|\Sigma|$ operation , but one could choose any reversible operation $\oplus$. This version is called *qubit-query*. In the Boolean case, when $\Sigma = \{0,1\}$, we get $\mathcal{O}_x |i\rangle |z\rangle = |i\rangle |z \oplus x_i\rangle = (1 - x_i) |i\rangle |z\rangle + x_i |i\rangle |1 - z\rangle$.

Alternatively, the query operator can give the query outcome in the phase, which version is called *phase-query*. For this version, we identify each element of Σ with an integer from $[|\Sigma|]$ in a bijective manner. The phase query operator acts as $\mathcal{O}_x^{\pm} |i\rangle |z\rangle = \omega^{z \cdot x_i} |i\rangle |z\rangle$, for $z \in \Sigma$ and $i \in I$, where ω is the $|\Sigma|$-th root of unity, i.e. $\omega = e^{i2\pi/|\Sigma|}$. For example, if $x \in \{0,1\}^n$ and the initial state is the uniform superposition over the indices then $\mathcal{O}_x^{\pm} \sum_{i \in [n]} |i\rangle |1\rangle = \sum_{i \in [n]} (-1)^{x_i} |i\rangle |1\rangle$. So, in the resulting state exactly those indices get phase -1 where x contains a 1-bit. One can show that the two quantum query models are equivalent, i.e. one type of query can be simulated by the other type.

Quantum memory models

The memory model is an important question in any computational model. In classical computing, the algorithm is executed by the CPU and the memory, RAM, is another device with a different physical implementation. For various applications, quantum computers need a similar but quantum memory, usually called *QRAM*. For a survey on QRAM see [JR23], below we shortly mention some important aspects.

Both the data stored and the access to it (addressing) can be either classical or quantum. Classical Random-Access Classical Memory (CRACM) yields the usual classical RAM. Classical Random-Access Quantum Memory (CRAQM) corresponds to a classical control of the qubits. From the memory point of view the two most relevant versions are the ones with quantum access. In the case of Quantum Random-Access Quantum Memory (QRAQM), the data is stored in a quantum register and the algorithm can read and write it in superposition. In this work, we do not need QRAQM.

Quantum Random-Access Classical Memory (*QRACM*) is the variant we are going to use in this thesis. For quantum algorithms that solve classical problems, it usually suffices to store classical information about the input in their memory. But for leveraging quantum phenomena, it is necessary to have quantum access to it (i.e. to be able to address data in superposition). This means that the data is stored in a classical table T: for example, in the case of binary n-bit data, $T \in \{0,1\}^n$. It is addressed by a quantum register of $\lceil \log n \rceil$ qubits, and there is an additional qubit for the output. QRACM means access to a unitary U_T such that $U_T \left|i\right\rangle \left|b\right\rangle = \left|i\right\rangle \left|b \oplus T_i\right\rangle$. In fact, this is exactly what we assume in the case of quantum query complexity: the physical implementation of the input oracle would be a QRACM.

Similarly to classical RAM, we usually assume that QRAM access takes negligible time compared to other steps of an algorithm. Unfortunately, this assumption is not yet experimentally justified.

2.5.4 The polynomial method

The polynomial method is a lower bound technique for quantum query complexity, introduced in [BBC+01]. In its simplest version, we have a Boolean alphabet (i.e. $\Sigma = \{0,1\}$), and the algorithm has to compute a total function $f : \{0,1\}^n \to \{0,1\}$ with high probability for any input $x \in \{0,1\}^n$, and the algorithm can use the query operator $\mathcal{O}_x$.

Definition 2.5.1 (approximate degree)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ and $\varepsilon > 0$. A polynomial $p : \{0,1\}^n \to \mathbb{R}$ ε-*approximates* f if $\forall x \in \{0,1\} : |f(x) - p(x)| < \varepsilon$. Moreover, the ε-*approximate degree* $\deg_\varepsilon(f)$ *of* f is the smallest degree of such a polynomial.*

Using the following result, it is possible to prove lower bounds on the quantum query complexity of several problems – we are going to use it in Section 4.4. We give an overview of the proof.

Theorem 2.5.2. *Let $f : \{0,1\}^n \to \{0,1\}$. If a quantum algorithm, having query access to any input $x \in \{0,1\}^n$, computes $f(x)$ with error probability at most 1/3, then the algorithm needs to make at least $\deg_{1/3}(f)/2$ many queries.*

Proof sketch. Using the deferred measurements principle, every quantum query algorithm can be represented like the circuit in Figure 2.3: first it performs some input-independent

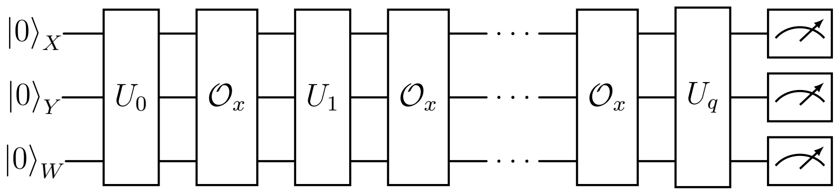

Figure 2.3 – The circuit of a quantum query algorithm.

operations represented by a unitary U_0, then it makes a query by using unitary $\mathcal{O}_x$, then some further input-independent operations U_1, etc. The circuit has three registers: X of n qubits for the query, Y of a single qubit for the answer, and the other qubits that we call the work register W. From now on, we omit the work register.

The state of the algorithm just before making the $(t+1)$-th query is $|\psi_t\rangle = U_t\mathcal{O}_xU_{t-1}\mathcal{O}_x\dots U_0|0,0\rangle$. At the beginning, $|\psi_0\rangle = U_0|0,0\rangle$ does not depend on input x because at this stage no query has been made yet. Let us say that $|\psi_0\rangle = \sum_{j=0}^{2^n-1}\sum_{z=0}^{1}\alpha_{j,z}^{(0)}|j,z\rangle$, where the amplitudes $\alpha_{j,z}^{(0)}$ are degree-0 polynomials in the x_i variables. For induction, let us assume that after t queries it is still true that in $|\psi_t\rangle = \sum_{j=0}^{2^n-1}\sum_{z=0}^{1}\alpha_{j,z}^{(t)}|j,z\rangle$ all the amplitudes $\alpha_{j,z}^{(t)}$ are polynomials of degree at most t. Then let us look at what happens when we apply the query operator on $|\psi_t\rangle$.

$$\mathcal{O}_x|\psi_t\rangle = \sum_{j=0}^{2^n-1}\sum_{z=0}^{1}\alpha_{j,z}^{(t)}\mathcal{O}_x|j,z\rangle = \sum_{j=0}^{2^n-1}\sum_{z=0}^{1}\alpha_{j,z}^{(t)}((1-x_j)|j,z\rangle + x_j|j,1-z\rangle)$$

By the induction hypothesis, in timestep t every $\alpha_{j,z}^{(t)}$ is a polynomial of degree at most t in the x_i variables, and by the above expression, the new amplitudes' degrees only increase by one. Since unitary U_{t+1} does not depend on the input, it cannot increase this degree. We can conclude that if the algorithm makes q queries, then the degree of its amplitudes in the state before the measurement is a polynomial of degree at most q. Since the algorithm's output depends on some measurement outcomes on this state, the probability of outputting any bit, say 1, is a polynomial p of degree at most $2q$ because of the Born rule.

Now if this q-query algorithm computes f with probability at least $2/3$ on every input x, then this polynomial $p(x)$ has to be close to $f(x)$ for every $x \in \{0,1\}^n$: in particular, p is 1/3-approximating polynomial of f. Thus, since we know that any 1/3-approximating polynomial of f has degree at least $\deg_{1/3}(f)$, it follows that the quantum query complexity of f is at least $\deg_{1/3}(f)/2$. □

Example 2.5.3. *Let $f\{0,1\}^n \to \{0,1\}$ be the PARITY function: it is 0 if the input contains an even number of 1s and it is 1 otherwise. Since the problem is symmetric, i.e. it is invariant under any permutation of the input bits, it suffices to only consider the Hamming weight of the input x: we denote $f' : [n]_0 \to \{0,1\}$ the symmetrised version of f. Instead of looking at polynomials over all the x_i, we can use the symmetrised version of those too: this way the degree can only decrease, so a lower bound on the symmetric version's degree implies a lower bound on the original degree.*

It is clear that f' is alternating: $f'(0) = 0, f'(1) = 1, f'(2) = 0, f'(3) = 1, \dots$. Consequently, any approximating polynomial p of f' needs to satisfy $p(0) \leq 1/3, p(1) \geq 2/3, p(2) \leq 1/3, p(3) \geq 2/3, \dots$. Because of this fluctuation, polynomial $p - 1/2$ has n

zeroes, thus p has degree at least n. Using the polynomial method (Theorem 2.5.2), a quantum query complexity lower bound of $n/2$, i.e. $\Omega(n)$ follows.

2.5.5 Grover's algorithm

One of the most well-known quantum algorithms is Grover's algorithm for unordered search (or Grover search) [Gro96]. Here the input is a list of n elements, t of which are "marked", and one has to find a marked element. An alternative, decision version of the problem is binary OR with a promise: given as input a Boolean function $f : D \to \{0,1\}$ with $|D| = n$, that is promised to satisfy either $|f^{-1}(1)| = t$ or $|f^{-1}(1)| = 0$, the task is to decide which is the case, i.e. if there is an $x \in D$ such that $f(x) = 1$. Classically, the complexity of solving this problem with high (constant) probability is $\Theta(n/t)$, but quantumly it is $\Theta(\sqrt{n/t})$ [Gro96, BBHT99]. This remains true in expectation even when the number of solutions t is not known in advance.

We are going to use a particular variant of this result, that has been used many times in the literature (see e.g., [Amb04, Item 3 in Section 2.2], which was implicitly proved in [BBHT99]).

Theorem 2.5.4. *Let $1 \leq t_0 \leq N$. There exists a quantum algorithm that, given t_0 and query access to any function $f : D \to \{0,1\}$, makes $O(\sqrt{N/t_0})$ queries to f and outputs either "not found" or an element uniformly at random in $f^{-1}(1)$. Moreover, when $|f^{-1}(1)| \geq t_0$ the later occurs with high constant probability.*

Remark 2.5.5. *In practice, we will use this theorem when querying $f(x)$, for $x \in D$, requires making c queries to the input. In that case the total query complexity to G is $O(c\sqrt{N/t_0})$.*

Let us look at two modifications of this problem.

- If the number of marked elements is t, and an upper bound $t_1 \geq t$ is known, then finding all the marked elements with probability 1 takes $\Theta(\sqrt{nt_1})$ queries. It is a well-known result, to the best of our knowledge it was first formally proved in [dGdW02, Lemma 2]. See also the recent work of [vAGN24], where the authors make a tight resource analysis of this algorithm.
- Finding a minimum (or a maximum) of a set of n elements with high probability can be done in $\Theta(\sqrt{n})$ queries [DH96].

An important generalisation of Grover search is called amplitude amplification, where we are given black-box access to an algorithm $\mathcal{A}$ that returns a state $|\psi\rangle$. This state $|\psi\rangle$ is a superposition over a set of elements some of which are "good", and if $|\psi\rangle$ is measured, the probability of a good outcome is p. Classically, if we want to boost the probability of getting a good output to a constant (say 2/3) then we have to repeat $\mathcal{A}$ $\Theta(1/p)$ times. Quantumly, as a generalisation of Grover search, it suffices to repeat it $\Theta(1/\sqrt{p})$ times [BHMT02]. We formalise this in the next theorem statement.

Theorem 2.5.6. *Let $\mathcal{A}$ be a quantum algorithm that does not make any measurements and returns a state $|\psi\rangle$ that, when measured, the probability of a "good" outcome is $p > 0$. Then there is another quantum algorithm that, not knowing the value of p and having black-box access to $\mathcal{A}$ and its inverse, makes $O(1/\sqrt{p})$ calls to $\mathcal{A}$ and its inverse, and returns a state $|\psi'\rangle$ that, when measured, the probability of a good outcome is 2/3.*

Grover's algorithm, minimum finding and amplitude amplification are used as a subroutine in many quantum algorithms, because search is ubiquitous in algorithm design. Since the quadratic speedup of search is tight by the lower bound of [BBHT99], most problems admit at most quadratic quantum speedup in some of their subprocedures. Later in the thesis we are also going to use Grover search as a subroutine.

Just like any quantum algorithm, Grover search and its variants have some bounded (at most some constant) error probability. Because of this, when they are used as a subroutine, sometimes even in a nested way, the error can accumulate and make the error probability of the overall algorithm large. To prevent this from happening, one can reduce the error probability of each Grover instance to inverse polynomial by repeating it several times. The overhead is maximum a factor of $\log(1/\delta)$ to achieve error probability at most δ, and this overhead counts as negligeable in this thesis. Thus, from now on we assume that Grover search and its variants return a solution in $\tilde{\Theta}(\sqrt{n/t})$ queries without error.

2.6 Simplicial complexes and Betti numbers

An *(abstract) simplicial complex* over a set V of vertices is a set family of subsets of V, that is downward closed under the subset relation. It is called abstract because it is a purely combinatorial object with no associated geometry, in contrast with a geometric simplicial complex. In this thesis, we focus on finite abstract simplicial complexes, where the vertex set V is finite. A simplicial complex can be thought of as a higher-dimensional generalisation of graphs, or as a restricted hypergraph with the additional downward closedness constraint.

Definition 2.6.1 (Simplicial complex)**.** *A simplicial complex K is a collection of finite subsets of a vertex set V, such that if $S \in K$ and $S' \subset S$ then $S' \in K$. The sets in K with cardinality $k+1$ are called the k-faces or k-simplices of K.*

One can draw a *geometric realisation* of an abstract simplicial complex in some space, for example in a Euclidean space of appropriate dimension as follows. The vertices are represented (drawn) by points, the edges by line segments between the corresponding points etc. In general, a simplex is represented by the convex hull of the points that represent the vertices of the simplex, and these points are in general position. In a geometric realisation of a complex, we require that the drawings of any two simplices intersect only at the drawing of the simplex that is their intersection.

Example 2.6.2. *An example of a simplicial complex's geometric realisation can be seen in Figure 2.4. Its vertices (or 0-faces) are labelled by capital letters. The edges (1-faces) are symbolised by segments between pairs of vertices. There are two triangles (2-faces) in the complex symbolised by the areas of grey colour: ABC and DEF.*

Triangle BEC could be added easily to the complex. But for adding triangle DFG one has to add edge FG too in order to maintain downward closedness.

We denote the set of k-faces of a complex K by $F_k(K) = \{S \in K, |S| = k+1\}$ and its size by $d_k(K) = |F_k(K)|$. When it is clear from the context which simplicial complex is being considered, we will write only F_k and d_k. The *dimension* of K is the largest integer k such that K contains at least one k-face, that is, $\dim(K) = \max\{k : d_k > 0\}$. The

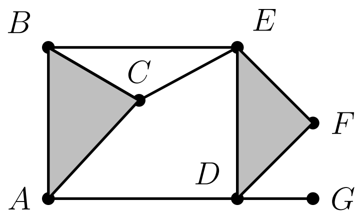

Figure 2.4 – A small example of a simplicial complex, see Example 2.6.2.

k*-skeleton* of K is the simplicial complex we get from K by deleting all its simplices of dimension larger than k. Notice that the 1-skeleton of a simplicial complex is a (simple) graph.

A *clique complex* is a special case of a simplicial complex and is defined by some underlying graph G. The sets in the clique complex associated to G are exactly the cliques of G. This implies, for instance, that a size-$(k+1)$ subset $S \subseteq V$ is in the complex if (and only if) all the size-k subsets of S are in the complex. For example, the simplicial complex of Figure 2.4 is not a clique complex because all the edges are present between vertices B, E and C but the triangle BEC is not included.

A *Vietoris-Rips complex* or flag complex is a special case of a clique complex. Here the vertex set corresponds to points (vectors) in a metric space, and two vertices are connected by an edge if their distance is less than a scaling parameter ϵ. Vietoris-Rips complexes are useful for applications because it can be a straightforward way of obtaining a clique complex from a data set with the data points corresponding to the vertices.

2.6.1 Orientation

Each simplex of the complex is assigned one out of two possible *orientations* that corresponds to the ordering of its vertices up to an even permutation. That is, if one can get one ordering from another by swapping pairs of elements an even number of times, then the two orderings correspond to the same orientation. For example, in a triangle ABC, the two possible orientations are $\{ABC, BCA, CAB\}$ and $\{ACB, CBA, BAC\}$. We say that $ABC, BCA, CAB, ACB, CBA, BAC$ all correspond to the same simplex, the first 3 of them with one orientation, and the rest with the opposite orientation. The two orientations are often described by the sign of the permutation, for example $\mathrm{sgn}(ABC) = \mathrm{sgn}(BCA) = -\mathrm{sgn}(ACB)$. This example is depicted on Figure 2.5, but in higher dimensions it is more difficult to have a similar figure because cyclic-permutation invariance is lost.

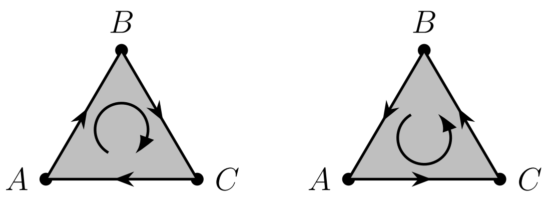

Figure 2.5 – The two orientations of a triangle and the induced orientations of the edges.

From now on, we take the vertex set $V = [n]$ (in the examples $V = \{A, B, C\}$) and the positive orientations are the ones corresponding to the increasing order of the vertices in each simplex. This way, we associate to each simplex a basis element (vector) $|S\rangle := |v_0 \dots v_k\rangle$, where $v_0 < v_1 < \dots < v_k$. If the orientation of our simplex is the opposite of the one corresponding to this ordering, then the associated element is $-|S\rangle$. Thus multiplication by -1 just flips the orientation.

The orientation of a simplex induces an orientation of all the smaller simplices it contains. In particular, having a k-simplex with associated element $|v_0 \dots v_k\rangle$, the induced orientation of its $(k-1)$-faces that we obtain by omitting a vertex with even index stays positive, and the others get a negative sign (using zero-based indexing). We can see an example of this in Figure 2.5: on the left drawing, simplex ABC with positive orientation induces orientations BC, $-AC = CA$ and AB of the edges.

2.6.2 Chain groups and boundaries

A *free module* is a generalisation of a vector space, where the coefficients (scalars) can come from a ring instead of a field. In particular, a free module on basis B, over ring $\mathbf{R}$ is the formal linear combinations of the elements of B with coefficients from $\mathbf{R}$, i.e. $\{\sum_{b\in B} \alpha_b b\}$ where $\alpha_b \in \mathbf{R}$. The *rank* of a free module is the size of its basis. A *free Abelian group* is a free module over the ring of integers $\mathbf{R} = \mathbb{Z}$.

The k*-chain* is a free module that consists of all possible linear combinations of the k-simplices of our complex. More formally, for simplicial complex K and $k \geq 0$, the k-chain C_k^K of K over a commutative ring $\mathbf{R}$ is defined as the formal linear combination of the k-faces of K, that is, $C_k^K = \{\sum_{i=1}^{d_k} \alpha_i |S_i\rangle\}$ where $S_i \in F_k(K)$ and $\alpha_i \in \mathbf{R}$. We will often write C_k instead of C_k^K when the complex does not need to be made explicit. If $\mathbf{R} = \mathbb{Z}$ then C_k is a free Abelian group, and if $\mathbf{R}$ is a field then it is a vector space.

Usually $\mathbf{R}$ is taken to be the set of integers $\mathbb{Z}$ or a field like $\mathbb{Q}, \mathbb{R}$ or $\mathbb{C}$. In the special case when $\mathbf{R} = \mathbb{F}_2$, the coefficients are either 0 or 1, and thus we have *unoriented* faces.

For simplicial complex K and each $k > 0$ integer, the k-th boundary operator ∂_k^K is a homomorphism that maps a k-face vector of K to the signed sum of the vectors of the $(k-1)$-faces that "surround" the k-face in K. It is defined formally as follows.

Definition 2.6.3 (Boundary operator). *Let K be a simplicial complex and $k \geq 1$ an integer. The k-th boundary operator of K is a homomorphism $\partial_k^K : C_k^K \to C_{k-1}^K$. For $S \in F_k(K)$ and $|S\rangle = |v_0, v_2, \dots, v_k\rangle$ it is defined by $\partial_k^K(|S\rangle) = \sum_{i=0}^{k} (-1)^{i-1} |S \setminus \{v_i\}\rangle$, and it extends to C_k^K by linearity.*

We will usually omit the complex from the notation and only write ∂_k.

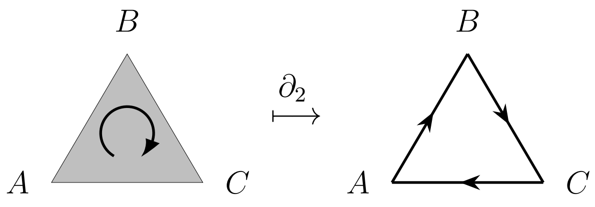

Figure 2.6 – The boundary of a triangle, see Example 2.6.4.

Example 2.6.4. *Let us use the definition of the boundary operator to show that Figure 2.6 is a correct depiction of it. If the vector associated to our triangle is $|ABC\rangle$. Then*

$$\partial_2(|ABC\rangle) = (-1)^0 |BC\rangle + (-1)^1 |AC\rangle + (-1)^0 |AB\rangle .$$

Since the negative sign means changing the orientation, we get $|BC\rangle+|CA\rangle+|AB\rangle$, matching Figure 2.6.

We can also see that

$$\partial_1(\partial_2(|ABC\rangle)) = \partial_1(|BC\rangle+|CA\rangle+|AB\rangle) = (|C\rangle-|B\rangle)+(|A\rangle-|C\rangle)+(|B\rangle-|A\rangle) = 0.$$

2.6.3 Homology groups and Betti numbers

In the example, we saw that the boundary of the boundary was zero. This is also true in general, meaning $\partial_k \circ \partial_{k+1} = 0$ for any simplicial complex. This means that $\operatorname{im}(\partial_{k+1}) \subseteq \ker(\partial_k)$ which implies that the quotient module $\ker(\partial_k)/\operatorname{im}(\partial_{k+1})$ is well-defined, and it is usually called the k-th *homology group* of the simplicial complex. Now we can define the Betti numbers, that are at the centre of interest in Chapter 3.

Definition 2.6.5 (Betti number). *Let $k \geq 0$ integer. The k-th Betti number β_k^K of a simplicial complex K is the rank of the k-th homology group:*

$$\beta_k^K = \text{rk}(\ker(\partial_k^K)/\operatorname{im}(\partial_{k+1}^K)).$$

Again, we will usually write β_k.

The elements of the k-th homology group are equivalence classes, and we define a k-dimensional *hole* as a member of an equivalence class of the k-th homology group. More intuitively, a hole is an element $|H\rangle$ of the k-chain C_k that has no boundary (i.e. its boundary is 0) and is no boundary. Equivalently, the following two constraints hold for a hole.

- It is a *cycle*, formally $\partial_k(|H\rangle) = 0$, meaning $|H\rangle \in \ker(\partial_k)$;
- there exists no $|H'\rangle \in C_{k+1}$ such that $\partial_{k+1}(|H'\rangle) = |H\rangle$, so $|H\rangle \notin \operatorname{im}(\partial_{k+1})$.

Then β_k counts the number of "independent" k-dimensional holes in the complex, that is, the number of equivalence classes of holes. We will give a combinatorial way to formalise this notion of independence in Chapter 3, using matroids.

Let us look at what Betti numbers mean intuitively in low dimensions. In the special case of $k = 0$, β_0 counts the number of connected components in the complex. The first Betti number β_1 counts those cycles made of edges, that are not filled by triangles. The second Betti number counts those cavities made of triangles, that are not filled by tetrahedra.

Example 2.6.6. *We continue the previous example to conclude that in a simplicial complex over 3 vertices A, B, C, expression $|H\rangle = (|BC\rangle + |CA\rangle + |AB\rangle)$ is a 1-dimensional hole if and only if triangle $|ABC\rangle$ is not in the complex. Indeed, we saw that $|H\rangle \in \ker(\partial_1)$, and also that $|H\rangle \in \operatorname{im}(\partial_2)$ if $|ABC\rangle$ is in the complex. Since in a complex with 3 vertices there cannot be another triangle, we are done.*

Torsion in the homology group

The homology groups and Betti numbers depend on the choice of the coefficient ring $\mathbf{R}$. In particular, according to the universal coefficient theorem, one can determine the homology groups under different choices of $\mathbf{R}$ from the homology group with $\mathbf{R} = \mathbb{Z}$, that can be written a direct sum of two parts, called the free part and the torsion part. For simplifying calculations, $\mathbf{R}$ is usually taken as a field, and next we focus on this case.

When the coefficient ring is a field When the coefficients are taken from a field of finite characteristic, for example $\mathbf{R} = \mathbb{F}_2$, then the torsion part in the homology group may still be nonzero, but in some cases, it can become zero. In the case where the coefficients are taken from a field of characteristic 0 (e.g. $\mathbf{R} = \mathbb{R}$), then the torsion part of the homology group is zero, and the free part embeds into a vector space. Thus, in this case, in the definition of the Betti numbers we can write dimension instead of rank. In fact, Betti numbers are often defined this way, as the dimension of the free part of the homology group [New18, Hat02]. In this thesis we use Definition 2.6.5 that does not constrain the choice of the coefficient ring, and thus it depends on this choice.

When there is a Euclidean geometric realisation In most applications, the simplicial complex can be "drawn" onto a low dimensional Euclidean space. As a reminder, a complex has a geometric realisation in a space if it is possible to "draw it nicely" in that space; and the k-skeleton of a simplicial complex K is the subcomplex of all the at-most-k-dimensional simplices of K. One can show that for a given $k \geq 0$, if the $(k+1)$-skeleton of the complex has a geometric realisation in the $(k+1)$-dimensional Euclidean space $\mathbb{R}^{k+1}$, then the k-th homology group over $\mathbb{Z}$ has zero torsion, and thus the k-th Betti number is the same under different choices of $\mathbf{R}$ [Jon].

Combinatorial Laplacian

For a simplicial complex K and an integer $k \geq 0$, the k-th *combinatorial Laplacian* $\Delta_k^K : C_k^K \to C_k^K$, usually written as Δ_k, is defined as

$$\Delta_k = \Delta_k^{\downarrow} + \Delta_k^{\uparrow},$$
$$\text{with} \quad \Delta_k^{\downarrow} = \partial_k^* \circ \partial_k \quad \text{and} \quad \Delta_k^{\uparrow} = \partial_{k+1} \circ \partial_{k+1}^*.$$

Here ∂_k^* is the adjoint boundary operator (also called the coboundary operator) that, written as a matrix in the standard basis, is just the transpose of the usual boundary operator. For example, as the boundary operator maps a triangle to the edges that surround it, the adjoint of the boundary operator maps an edge to the triangles of which it is an edge.

This way, the combinatorial Laplacian can be seen as a generalisation of the usual graph Laplacian. The graph Laplacian is $\Delta_0 = \Delta_0^{\uparrow}$ (since we cannot go to dimension -1, $\Delta_0^{\downarrow} = 0$): intuitively it maps a vertex to those vertices with which it shares an edge. In higher dimensions there are two options, for example we can map an edge to another if they share a common endpoint ($\Delta_1^{\downarrow}$), or if they are edges of the same triangle ($\Delta_1^{\uparrow}$).

Δ_k is a symmetric, positive semidefinite matrix of size $d_k \times d_k$. Therefore, its eigenvalues are all nonnegative, the smallest one is 0, and the smallest nonzero eigenvalue $\lambda_2(\Delta_k)$ is called the *spectral gap* of the Laplacian. The largest eigenvalue is denoted by $\lambda_{\max}(\Delta_k)$, and we call matrix $\Delta_k/\lambda_{\max}(\Delta_k)$ the k-th *normalised Laplacian*. The normalised Laplacian has eigenvalues between 0 and 1 and its smallest nonzero eigenvalue is $\gamma = \lambda_2/\lambda_{\max}$, called the *normalised spectral gap* of the Laplacian.

Based on Hodge theory [Hod41], we can get an equivalent definition of the Betti numbers using the combinatorial Laplacian:

Observation 2.6.7. *The k-th Betti number over fields of characteristic 0 (i.e. in the torsion-free case) $\beta_k = \dim\ker(\Delta_k)$, thus it is equal to the number of zero eigenvalues of Δ_k.*

2.6.4 Persistent Betti numbers and Laplacians

In particular, the so-called *persistent Betti numbers* have been useful for applications [PEv+16, KMH+21, BAD21], they were introduced in [ELZ02]. Here, usually the Vietoris-Rips complexes of a data set is considered at varying scaling parameters ϵ. Starting with $\epsilon = 0$, where all the vertices are isolated, ϵ is slowly increased which makes some vertices connected, and holes start to be formed. In the end, ϵ is so large that the graph becomes a clique, when all the holes are "filled". During this process, the persistent holes, those that remain holes for several values of ϵ, are considered to be more important, because they do not depend on the choice of ϵ too much: they capture a scale-independent global property of the data set.

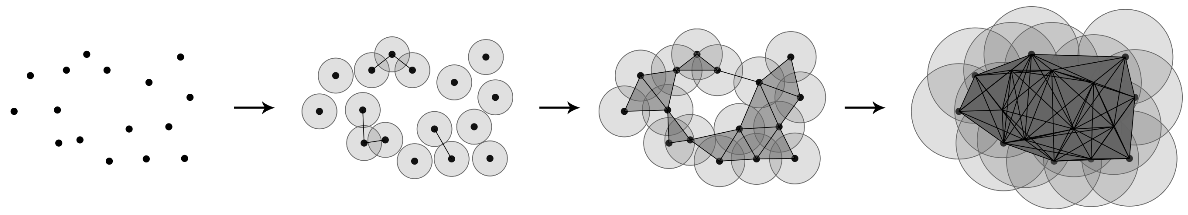

Figure 2.7 – Filtration: Vietoris-Rips complexes at different ϵ values. Source: [Rie17].

Now let us define persistent Betti numbers more formally. A *filtration* is a sequence of simplicial complexes over the same vertex set where each complex contains the previous ones: $K_1 \subseteq K_2 \subseteq \cdots \subseteq K_t$. For example, taking Vietoris-Rips complexes of a data set for several values of increasing scaling parameters ϵ defines a filtration.

Let us take any two simplicial complexes K, L over the same vertex set such that $K \subseteq L$ – for example K is the Vietoris-Rips complex of a data set at scaling parameter ϵ_1 and L is the one at $\epsilon_2 > \epsilon_1$. The k-th persistent Betti number corresponding to pair (K, L) is defined as

$$\beta_k^{K,L} = \mathrm{rk}(\ker(\partial_k^K)/(\mathrm{im}(\partial_{k+1}^L) \cap \ker(\partial_k^K))).$$

Alternatively, in the above definition $(\mathrm{im}(\partial_{k+1}^L) \cap \ker(\partial_k^K))$ can be replaced by $\mathrm{im}(\partial_{k+1}^{K,L})$, where $\partial_{k+1}^{K,L}$ is the *persistent boundary operator*, that is like the usual boundary operator ∂_{k+1}^L in L, but its image is restricted to the k-chain group C_k^K of K. This intuitively means that the "persistent boundary" of a $(k+1)$-face in L is a combination of k-faces in K. This makes sure that only those holes are counted that are holes both in K and L.

Persistent Laplacians can be defined analogously: $\Delta_k^{K,L} = (\partial_k^K)^*\partial_k^K + \partial_{k+1}^{K,L}(\partial_{k+1}^{K,L})^*$. It was shown in [MWW22] that $\beta_k^{K,L} = \dim\ker(\Delta_k^{K,L})$.

Chapter 3

Classical algorithms for Betti number estimation

3.1 Introduction

In this chapter, we use simplicial complexes to model data sets. As we mentioned in the previous chapters, Betti numbers are important features of simplicial complexes: intuitively they characterise the number of high dimensional holes and thus give information about the topology of the complex.

Unfortunately, computing Betti numbers efficiently seems like a challenging task, where by an efficient algorithm, we mean one with time complexity polynomial in the number of vertices n. Indeed, it was recently shown in [CK24] that multiplicatively approximating the Betti numbers of a simplicial complex is hard for quantum computers, even in the special case of clique complexes. In particular, without going into precise definitions of the complexity class, multiplicative approximation of Betti numbers is QMA1-hard, where QMA1 is the 1-sided error version of QMA, and QMA is the quantum analogue of NP.

3.1.1 Additive approximation of Betti numbers

The next natural question is whether we can *additively* approximate the Betti numbers. More formally, given a parameter $\varepsilon \in (0, 1)$, can we efficiently output an estimate $\tilde{\nu}_k$ of the k-th (normalised) Betti number of the complex satisfying

$$\tilde{\nu}_k = \frac{\beta_k}{d_k} \pm \varepsilon,$$

where d_k denotes the number of k-faces in the complex. To the best of our knowledge, Elek was the first to study this question. In [Ele10], it was proved that if the complex has constant degree, that is, every 0-face (vertex) of the complex is contained in a constant number of 1-faces (edges), then there is an algorithm whose running time depends only on the parameter ε. The constant degree assumption, however, implies that the complex has constant dimension as well (that is, it contains no k-faces for $k \in \omega(1)$), and thus all Betti numbers β_k for non-constant k are zero.

The problem was later reconsidered by Lloyd, Garnerone, and Zanardi [LGZ16], who proposed a *quantum* algorithm for estimating the Betti numbers. Their algorithm combines quantum techniques such as Hamiltonian simulation and quantum phase estimation. Assuming that we can efficiently sample a uniformly random k-face from the complex, the algorithm outputs an ε-additive approximation of β_k in time

$$\mathrm{poly}(n, 1/\gamma, 1/\varepsilon),$$

where $n = d_0$ denotes the number of 0-faces in the complex, and γ is the normalised spectral gap of the k-th combinatorial Laplacian.

As current classical algorithms for calculating Betti numbers seem to run in time $\mathrm{poly}(d_k)$ [Fri98], which can be $\mathrm{poly}(n^k)$, this suggests an exponential speedup (in k) of quantum algorithms over classical ones for this problem. This explains the surge of interest in the quantum algorithm, and in particular, in its application to clique complexes, which have a concise $\mathrm{poly}(n)$-size description [AUC+24, CK24, MGB22, BSG+24, SL23, AMS24, Hay22].

Our results: Section 3.3 is based on our results published in **[AGSS23]**, where we describe a simple classical algorithm for approximating Betti numbers using the path integral Monte Carlo method. Our algorithm provides a natural benchmark for the aforementioned quantum algorithms. Similarly to these quantum algorithms, our algorithm runs in polynomial time if the gap and the precision are "large". However, while the quantum algorithms can afford a gap γ and precision ε as small as $1/\mathrm{poly}(n)$, our algorithm requires these to be constant for general complexes. This is similar to the dequantization results from [GLG22].

In the case of clique complexes, we can go further. For example, if $k \in \Omega(n)$ then we can afford precision $\varepsilon = 1/\mathrm{poly}(n)$ if the gap is constant, or gap $\gamma = \Omega(1/\log^2 n)$ if the precision is constant. Overall, our result does not exclude a potential exponential quantum advantage for the problem of estimating Betti numbers, but it narrows down the region where it is possible. Below we give a more detailed overview of these results.

Technical overview

Using Hodge theory, we know that the k-th Betti number is the dimension of the kernel of the k-th combinatorial Laplacian Δ_k. Let us define matrix $H = I - \Delta_k/\hat{\lambda}$ where $\hat{\lambda}$ is an estimate of the maximum eigenvalue of Δ_k. We can relate the trace of a large enough power of H to the k-th Betti number: $\mathrm{tr}(H^r)$ is between β_k and $\beta_k + \varepsilon d_k$ if $r \geq \frac{\hat{\lambda}}{\lambda_2}\log(1/\varepsilon)$ where d_k is the number of k-faces in the complex and λ_2 is a lower bound on the spectral gap of Δ_k.

Now to estimate the k-th normalised Betti number β_k/d_k, it suffices to estimate the normalised trace of this large enough power of H: $\mathrm{tr}(H^r)/d_k = \frac{1}{d_k}\sum_{i=1}^{d_k} \langle i| H^r |i\rangle$. This expression is the expected value of a random variable Y_r defined by the following process. Take a k-face uniformly at random and take r steps on the Markov chain with transition probabilities corresponding to the elements of H. That is, the rows and columns of H correspond to the k-simplices of the complex, and if we are on the i-th simplex, the next one is going to be the j-th simplex with probability $|H_{i,j}|/\|H_{\cdot,i}\|_1$. The value of Y_r is an appropriate nonzero value (depending on the path taken) if at the end of this process we are back at the initial k-face, and 0 otherwise.

We show that it is possible to sample from Y_r in time $r \cdot \mathrm{poly}(n)$ by taking r steps on the Markov chain defined above, using the sparsity of H. Using a standard concentration bound

(Hoeffding's inequality), we can upper bound how many samples we need to take from Y_r to have a good approximation of its expectation with high probability. The complexity it yields depends on the 1-norm of H that we can upper bound using the sparsity of Δ_k: we get complexity $n^{O\left(\frac{1}{\gamma}\log\frac{1}{\varepsilon}\right)}$.

We can improve the complexity with the following trick. Using Chebyshev polynomials, any monomial of degree r can be approximated by a polynomial of degree roughly $\sqrt{r}$. This way, instead of directly approximating the normalised trace of H^r, we can approximate H^r with a polynomial of lower degree and do the previously explained process for each monomial of the polynomial. By calculating the combination of these estimates with the polynomial's coefficients, we obtain our desired approximation. This improves the complexity to $n^{O\left(\frac{1}{\sqrt{\gamma}}\log\frac{1}{\varepsilon}\right)}$.

The previous results work for general simplicial complexes. For the special case of clique complexes, we get an improved complexity by noticing that the combinatorial Laplacian of clique complexes is even sparser than in the general case: instead of $O(nk)$ nonzero elements, there are $O(n)$ many. This results in a better bound when using Hoeffding's inequality. In particular, we obtain complexity $(n/\hat{\lambda})^{O\left(\frac{1}{\sqrt{\gamma}}\log\frac{1}{\varepsilon}\right)} \cdot \mathrm{poly}(n)$.

Since $\hat{\lambda}$ is approximately the maximum eigenvalue of Δ_k, which is known to be lower bounded by k, in the high dimensional case, when $k \in \Omega(n)$ (which is the interesting case for the quantum algorithm of [LGZ16]), we get a polynomial runtime if $\frac{1}{\sqrt{\gamma}}\log\frac{1}{\varepsilon} \in O(\log n)$.

Open problems A natural question is to what extent we can improve our results. The most stringent barrier seems to come from [CC24, Theorem 6], who proved that Betti number estimation for general (not necessarily simplicial) complexes is DQC1-hard when $\varepsilon, \gamma = 1/\mathrm{poly}(n)$, where DQC1 is a complexity class that is expected to be hard to simulate classically. (In fact, they consider a slight generalization of the problem called "quasi-Betti number estimation".) This safeguards a quantum speedup for the case of general complexes, yet it leaves open the case of clique complexes. Our work shows that we can get additional leverage for clique complexes. We leave it as our main open question whether the classical complexity for clique complexes can be improved to $\mathrm{poly}(n, 1/\gamma, 1/\varepsilon)$.

The task of estimating persistent Betti numbers has been getting an increasing amount of attention, see e.g. [WNW20, MWW22, Hay22]. It seems like our method can be used for solving this problem too (if we have membership query access to both complexes K and L with $K \subseteq L$), but we leave the formal proof of this for future work. Moreover, the speedup in the clique complex case is kept because the persistent Laplacian can only get even sparser compared to the usual combinatorial Laplacians. In particular, Lemma 3.3.11 still holds, but with the up-degree in L that can only be larger than in K.

A final open question, as was already mentioned in earlier works [BSG+24], is characterizing which complexes admit a large spectral gap. The advantage of our algorithm, as well as the aforementioned quantum algorithms, hinges on this assumption. As we mentioned earlier, [BSG+24] discussed the complete k-partite graph $K(m, k)$ as an example where our spectral gap assumption on the combinatorial Laplacian holds: see our statement in Proposition 3.2.5. Observe that for $K(n/k, k)$, the spectral gap is n/k, and the normalised spectral gap is $\gamma = 1/k$. Thus, for this kind of complexes our algorithm runs in polynomial time if for example $\varepsilon \in \Omega(1)$ and $k \in O\left(\log^2(n)\right)$.

Related work. Upon completion of this work, we noticed that a similar classical path integral Monte Carlo algorithm for estimating Betti numbers was proposed in [BSG$^+$24]. The authors use a Trotterisation approach to implement an imaginary time evolution of the combinatorial Laplacian, and use a more complex distribution over paths to minimise the variance of the Monte Carlo estimator. These techniques seem more flexible than ours and might eventually lead to a better algorithm. However, unless some additional conditions are imposed, the current runtime of their algorithm still shows an exponential dependency on both k and $1/\varepsilon$, which our algorithm avoids.

As a reviewer has pointed out, a similar usage of the path integral Monte Carlo method to estimate elements of a matrix power is present in [DSTS17] (in particular, see their Lemma 2.5).

Since its publication, several articles have used our results, let us mention two of these. In [ABC$^+$24] the authors implement and compare our algorithms' performance to some other work, highlighting their similar Monte Carlo structure. Moreover, they slightly improve the estimation our algorithm provides, by using a different polynomial for approximating the matrix power. They also provide a new quantum algorithm by combining our classical method with the quantum algorithm of [AUC$^+$24].

The work of [CWS$^+$25] investigates the possibility of a superpolynomial quantum speedup for computing matrix functions. They use some of our results in order to see what is "easy" for classical algorithms.

3.1.2 Property testing Betti numbers

Our work presented in Section 3.4 investigates this question from a new, property testing perspective in the dense graph model, and it yields a tool to investigate whether typical graphs can have a (very) large Betti number. This is relevant for some of the previously mentioned algorithms, since an additive estimate of a normalised Betti number is only interesting if the Betti number is large.

The result of Elek [Ele10] in the bounded-degree model can also be viewed from a property testing perspective. In their model, the graph is assumed to have a constant bound d on the vertex degrees, and a query reveals the (at most d) neighbours of a vertex. Elek showed that, for any $\varepsilon > 0$ and with a number of queries only dependent on ε, it is possible to return an estimate $\hat{\beta}_k$ satisfying $\hat{\beta}_k = \beta_k \pm \varepsilon n$ for $k < d$ (for $k \geq d$ necessarily $d_k = \beta_k = 0$). The proof is based on (sparse) graph limits and is completely different from our approach.

Unfortunately, such a result is not possible in the dense graph model: returning an estimate $\hat{\beta}_k = \beta_k \pm \varepsilon n$ (or even $\hat{\beta}_k = \beta_k \pm \varepsilon d_k$) requires $\Omega(n)$ many queries. To see this, consider the case $k = 0$ for which $d_0 = n$ and β_0 equals the number of connected components of the graph. The cycle graph has $\beta_0 = 1$, while any graph with $\leq n/2$ edges has $\beta_0 \geq n/2$. However, it takes $\Omega(n)$ queries to distinguish these graphs in the dense model. This motivates the weaker formulation of large Betti number testability that we use in Section 3.4. Note also that the contrapositive, having a small Betti number, is trivial to test. E.g., a graph cannot be far from having small Betti number β_0 since we can always add a cycle, thereby setting $\beta_0 = 1$.

Our results: In Section 3.4, based on our article **[SA25]**, we use the lens of (graph) property testing to further our understanding of Betti number estimation. In particular, we are given query access to the adjacency matrix of a dense graph G and we want to test if the k-th Betti number β_k of the clique complex associated to G satisfies $\beta_k \geq (1-\delta)d_k$. We show that this property (over $\mathbb{F}_2$) for constant k is testable with a constant number of queries in the dense graph model if δ is very small. More specifically, we prove that for any $\varepsilon > 0$, there exists $\delta(\varepsilon, k) > 0$ such that testing whether $\beta_k \geq (1-\delta)d_k$ for $\delta \leq \delta(\varepsilon, k)$ reduces to tolerantly testing $(k+2)$-clique-freeness, which is known to be testable. For this, we use a combinatorial understanding of simplicial complexes.

Technical overview

For proving the above result, first we observe that there is a notion of independence of k-simplices that is useful in the context of this work. This notion comes from matroid theory: we can assign a vector to each simplex based on its boundary, and a set of k-simplices is independent iff the corresponding vectors form a linearly independent set. On an intuitive level, this means that a set of k-faces is independent if no subset of them forms a k-dimensional hole. We denote the maximum size of an independent set of k-faces in the complex as r_k.

We show that r_k cannot be much smaller than the total number of k-faces d_k, in particular, $d_k(k+1)/n \leq r_k$. Then we present two proofs[1] of an elegant formula, which links the k-th Betti number to the total number of k-faces and to the maximum number of independent k- and $(k-1)$-faces: $\beta_k = d_k - r_k - r_{k-1}$.

Then we turn to proving our main result, first in the special case of $k = 0$. Using the previous formula, $\beta_0 \geq (1-\delta)d_0$ (with the number of vertices $d_0 = n$) is equivalent to $r_1 \leq \delta n$, i.e. it suffices to test if G has few independent edges. Then we use the fact that r_1 can be at most about a factor-n smaller than d_1, which leads to the conclusion that $\beta_0 \geq (1-\delta)d_0$ implies G to have few edges, and thus to be close to edge-freeness. Moreover, one can check that being far from $\beta_0 \geq (1-\delta)d_0$ implies being far from edge-freeness. Hence, for appropriate parameters ε, δ we reduced the problem of property testing very large β_0 to tolerantly testing edge freeness.

In general, for constant k we can reduce testing $\beta_k \geq (1-\delta)d_k$ to tolerantly testing $(k+2)$-clique-freeness, and the first part of the reduction is very similar to the $k = 0$ case. But for the "farness" part of the reduction, we need a slightly more complicated argument. By contraposition, we assume that there is a graph that is ε-far from having a very large β_k, but it is $\varepsilon/2$-close to $(k+1)$-clique-freeness. To reach contradiction, we use a construction that, given a graph H with few $(k+1)$-cliques and a proximity parameter α, provides another graph H' that is α-close to H and whose clique complex has a large β_k. The construction takes αn vertices of H and modifies the subgraph induced by this set to change it into a complete $(k+1)$-partite graph.

Moreover, in the reduction we need to use the fact that containing few $(k+2)$-cliques implies being close to $(k+2)$-clique-freeness, which is true by the well-known graph removal lemma. However, using the graph removal lemma requires our parameter δ to be very small, upper bounded by one over a tower function of $\log(1/\varepsilon)$ (this best known bound is due to [Fox11]). For the special cases of $k = 0$ and $k = 1$ we can use simple observations to avoid using the graph removal lemma and get much better bounds on δ.

1. Let us note that first we gave a combinatorial proof of this result, then we discovered that a very different, algebraic proof had already existed in the literature.

Unoriented faces We defined the k-chain group as the free module $C_k = \{\sum_{i=1}^{d_k} \alpha_i S_i\}$ where $\mathbf{R}$ is a commutative ring, $S_i \in F_k(K)$ and $\alpha_i \in \mathbf{R}$. As we discussed before, many sources consider integer or real coefficients ($\mathbf{R} = \mathbb{Z}$ or $\mathbf{R} = \mathbb{R}$), but for our combinatorial interpretation of homology, it is more natural to pick binary coefficients $\mathbf{R} = \mathbb{F}_2$. Homology over $\mathbb{F}_2$ is based on *unoriented* faces in which case the chain group $C_k = 2^{F_k}$ is simply the set of all subsets of F_k. We note that this way, the homology groups can have torsion, and thus the Betti numbers can change compared to the torsion-free case, but in most applications, this does not happen (see the discussion about torsion in Section 2.6.3). With this choice of binary coefficients, we will refer to the elements of C_k either as a sum of k-faces or as a set of k-faces – the two are equivalent.

Open questions. Our work raises a few open questions. The most obvious one is whether our results can be pushed further. For instance, it might be possible to test more moderately sized Betti numbers, or Betti numbers for non-constant k (the case of interest for quantum algorithms). Having similar results under different coefficient rings $\mathbf{R}$ is another perspective.

A final open direction is to introduce the framework of property testing abstract simplicial complexes, generalising graph property testing. By limiting ourselves to clique complexes, we could phrase our results in the graph property testing language, but this might not be the most natural approach.

3.2 Notations and preliminaries

3.2.1 Combinatorial Laplacians

Let us keep in mind the definitions of simplicial complexes and combinatorial Laplacians from Section 2.6. We define the degree and the neighbourhood of a face as follows.

Definition 3.2.1. *In a simplicial complex, the* up-degree *of a k-face S is the number of $(k+1)$-faces that contain S. It is denoted as $d_S^{up} := |\{S' \in F_{k+1} \text{ s.t. } S \subseteq S'\}|$. The maximum up-degree among all the k-faces is denoted as $\delta_k = \max_{S \in F_k} d_S^{up}$.*

Definition 3.2.2 (Down-up and up-down neighbours). *Let $S_1, S_2 \in F_k$ be two k-faces of a simplicial complex K. S_1 and S_2 are said to be* down-up neighbours *if their symmetric difference $|S_1 \triangle S_2| = 2$. Additionally, if $S_1 \cup S_2$ is a $(k+1)$-face of K, then S_1 and S_2 are also said to be* up-down neighbours.

The following lemma from [Gol02] uses these notions to characterise the entries of Δ_k.

Lemma 3.2.3 (Restatement of Laplacian Matrix Theorem, [Gol02, Theorem 3.3.4]). *Let K be a finite oriented simplicial complex, k be an integer with $0 < k \leq \dim(K)$, and $\{S_1, S_2, \ldots, S_{d_k}\} = F_k(K)$ denote the k-faces of K. Let $i, j \in [d_k]$. Then we have the following:*

- $(\Delta_k)_{ii} = d_{S_i}^{up} + k + 1$.
- $(\Delta_k)_{ij} = \pm 1$ *if $i \neq j$, and S_i and S_j are down-up neighbours but they are not up-down neighbours.*

– $(\Delta_k)_{ij} = 0$ *otherwise (i.e. if $i \neq j$, and either S_i and S_j are up-down neighbours or they are not down-up neighbours).*

The following lemma gathers some useful facts about Δ_k that will be used in our proofs.

Lemma 3.2.4. *Let us consider a simplicial complex K with Δ_k being its k-th combinatorial Laplacian and δ_k being the maximum up-degree among all k-faces of K. Then the following results hold:*

– $\delta_k + k + 1 \leq \lambda_{\max}(\Delta_k) \leq n$.

– $(\Delta_k)_{ii} \leq n$.

– Δ_k *has at most $(n-k-1)(k+1)$ nonzero off-diagonal entries in each row, all equal to ± 1* [2].

Proof. The second and third bullets follow from Lemma 3.2.3. The second inequality of the first bullet follows from [DR02, Proposition 6.2], who prove that $\lambda_{\max}(\Delta_k^{\uparrow}) \leq n$ and $\lambda_{\max}(\Delta_k^{\downarrow}) \leq n$. This gives the claimed bound if we use that $\lambda_{\max}(\Delta_k) = \max\{\lambda_{\max}(\Delta_k^{\uparrow}), \lambda_{\max}(\Delta_k^{\downarrow})\}$, which follows from $\Delta_k^{\uparrow}\Delta_k^{\downarrow} = \Delta_k^{\downarrow}\Delta_k^{\uparrow} = 0$ (which is true because $\partial_k(\partial_{k+1}(.)) = 0$).

For proving the first inequality of the first bullet, we write the largest eigenvalue using the Rayleigh quotient:

$$\begin{aligned}\lambda_{\max}(\Delta_k) &= \max_{\|x\|_2=1} x^T \Delta_k x = \max_{\|x\|_2=1} (x^T \Delta_k^{\uparrow} x + x^T \Delta_k^{\downarrow} x) \\ &= \max_{\|x\|_2=1} (\|\partial_{k+1}^* x\|_2^2 + \|\partial_k x\|_2^2).\end{aligned}$$

We can lower bound this by taking a particular x: the one that is all zero except for a position where it is one, and the latter position corresponds to a k-face with up-degree δ_k. For this vector, $\partial_{k+1}^* x$ contains δ_k ones and the other elements are zero (because it has up-degree δ_k). And it is also true that $\partial_k x$ contains $k+1$ ones and the other elements are zero (because every k-face contains $k+1$ many $(k-1)$-faces). This concludes that $\delta_k + k + 1 \leq \lambda_{\max}(\Delta_k)$. □

A clique complex with large Betti number and spectral gap

Some algorithms that estimate the k-th Betti number, including our algorithm presented in Section 3.3, require the spectral gap of the combinatorial Laplacian not to be too small, or even the Betti number to be large. In [BSG$^+$24, Section IV A] the authors describe a construction of a clique complex and prove that it satisfies both requirements. As it is relevant for us in this chapter, let us look at this result briefly.

For the $(k-1)$-st Betti number, the underlying graph of the clique complex is the complete k-partite graph $K(m,k)$, where the total number of vertices is $n = mk$. $K(m,k)$ consists of k clusters, where each cluster contains m vertices, and two vertices are adjacent iff they are in different clusters. The $(k-1)$-st combinatorial Laplacian of the clique complex defined by $K(m,k)$, has spectral gap m and the $(k-1)$-st Betti number of the complex is $(m-1)^k$ (see [BSG$^+$24, Proposition 1 & 2]). This implies the following proposition.

2. This is tight up to a constant for general simplicial complexes, in contrast to some earlier papers [GCD22, MGB22] that mention $\mathcal{O}(n)$ nonzero off-diagonal entries. Later in the chapter (see Lemma 3.3.11), we show that for clique complexes it is actually $\mathcal{O}(n)$, and we exploit this to obtain a more efficient algorithm.

Proposition 3.2.5. *Let $k < n$ be positive integers, such that n is an integer multiple of k. Then there is a clique complex on n vertices that has $(k-1)$-st Betti number $(n/k-1)^k$, and whose $(k-1)$-st combinatorial Laplacian has spectral gap n/k.*

3.2.2 Property testing subgraph freeness

The following well-known lemma (that can be proved using the Szemerédi regularity lemma [Sze78]) has been central to proving many testability results, and we will also use it in Section 3.4.

Lemma 3.2.6 (Graph removal lemma, [Für95])**.** *For any fixed graph H and any $\varepsilon > 0$, there exists a $\delta > 0$ such that the following holds: any n-vertex graph G ($|V(H)| < n$) that contains at most $\delta n^{|V(H)|}$ copies of H as subgraphs, can be made H-free by removing at most εn^2 edges (i.e. G is ε-close to being H-free).*

It follows almost directly from this result that, for any constant-sized graph H, the property of being H-free is testable, i.e. the query complexity of this property testing problem does not depend on n [ADL+94]. We note that the bound on δ in this result is extremely small, even using the improved bound of [Fox11]: $\delta = 1/\text{tower}(5|V(H)|^4 \log(1/\varepsilon))$, where $\text{tower}(1) = 2$ and for all $i \geq 1$, $\text{tower}(i+1) = 2^{\text{tower}(i)}$.

This can be combined with the fact that every testable property in the dense graph model is also *tolerantly* testable. More precisely, in [FN07] the authors prove that for every testable property there is a distance approximation algorithm, and this implies tolerant testability. This way, we get the following lemma which we are going to use later.

Lemma 3.2.7. *For any graph H, the property of H-freeness is tolerantly testable in the dense graph model. The number of queries depends only on the distance parameters $\varepsilon_1, \varepsilon_2$ and on $|V(H)|$.*

Remark 3.2.8. *The construction of [FN07] takes a tester for any property and builds a tolerant tester for the same property. The resulting query complexity is at least a tower in some function of the (non-tolerant) tester's query complexity. A later work [GKS23] obtained the following improved upper bound. If a property is testable for error parameter ε with query complexity $q(\varepsilon)$, then it is tolerantly testable with query complexity $2^{\text{poly}(1/\varepsilon)\cdot 2^{q(\varepsilon/2)}}$ ([GKS23] Theorem 9).*

3.2.3 Matroids

The appropriate notion of independence of simplices and of holes that we will need in Section 3.4, comes from matroid theory. A matroid is a downward closed set family with an additional property called the exchange property. In this sense, matroids are a specialisation of simplicial complexes, but we are going to use them in a different way.

Definition 3.2.9 (Matroid)**.** *A matroid M over ground set E is a family of subsets $I \subseteq 2^E$ called the independent subsets of E, and which satisfies the following properties.*

1. *$\emptyset \in I$.*
2. *If $A \in I$ and $B \subseteq A$ then $B \in I$.*
3. *If $A, B \in I$ and $|B| < |A|$ then $\exists v \in A \setminus B$ such that $B \cup \{v\} \in I$.*

The easiest example of a matroid is a graph. In this case, the ground set E in the matroid is the edge set of the graph, and we call a subset of edges independent if it is cycle-free. Matroids that can be defined this way by a graph are called *graphic matroids* or cycle matroids.

Another important example is linear independence of vectors. The elements of E are vectors from a vector space, and a subset of them is called independent if the vectors are linearly independent (over a field F). Matroids that can be defined in this way are called *linear matroids* (or representable over F).

By the *boundary vector* of a k-face $S \in F_k$, we mean a binary vector $\partial_k(S) \in \{0,1\}^{d_{k-1}}$, where a coordinate is 1 iff the corresponding $(k-1)$-face appears in the boundary of S (over $\mathbb{F}_2$). Note that notation ∂_k was defined as the boundary operator, but sometimes we are also going to use it to denote boundary vectors.

The *simplicial matroid* (or simplicial geometry) $M_k(K)$ associated to a simplicial complex K is a linear matroid defined as follows. It appears in e.g. [CR70, CL87].

Definition 3.2.10 (Simplicial matroid)**.** *The k-simplicial matroid $M_k(K)$ associated to a simplicial complex K is the linear matroid whose ground set is the set of boundary vectors $\partial_k(S) \in \{0,1\}^{d_{k-1}}$ for $S \in F_k(K)$.*

Motivated by this, we call a subset of k-faces independent if the corresponding boundary vectors are linearly independent (over the field $\mathbb{F}_2$).

A maximal independent set of a matroid M is called a *basis*. It is well known that all the bases of a matroid have the same size, equal to the *rank* $\mathrm{rk}(M)$ of the matroid. The full k-simplicial matroid $M_k(K_k^{\mathrm{full}})$ is the k-simplicial matroid associated to the full complex $K_k^{\mathrm{full}} = \{S \subseteq V, |S| \leq k+1\}$ that contains all the $k+1$-subsets as k-faces, but it does not have any higher dimensional face.

Proposition 3.2.11 (e.g. [CL87], Proposition 6.1.5)**.** $\mathrm{rk}(M_k(K_k^{\mathrm{full}})) = \binom{n-1}{k}$.

For a construction, fix a vertex u of K_k^{full} and take the set of k-faces that contain u. It is easy to see that this set of size $\binom{n-1}{k}$ is a basis of the matroid.

3.3 Additive approximation of Betti numbers

In a nutshell, we base our algorithm on a random variable whose expectation is close to β_k/d_k and whose variance is (sufficiently) small. Crucially, we show that we can efficiently generate samples from this random variable. Standard concentration bounds can be used to bound the required number of samples, and hence establish the complexity of our algorithm. More precisely, the algorithm is based on the technique of path integral Monte Carlo [Bar79], akin to the Ulam-von Neumann algorithm for solving linear systems [FL50]. Our result is formally stated below. By $\lambda_2(\Delta_k)$ we denote the *spectral gap* of the combinatorial Laplacian Δ_k, which is equal to its smallest nonzero eigenvalue.

Theorem 3.3.1. *Let Δ_k denote the k-th combinatorial Laplacian of the complex. Assume that in time* $\mathrm{poly}(n)$ *we can* (i) *draw a k-face uniformly at random, and* (ii) *check whether a set is in the complex. Given an estimate $\lambda_{\max}(\Delta_k) \leq \hat{\lambda} \leq c\lambda_{\max}(\Delta_k)$ for some constant $c > 0$ and a lower bound γ such that Δ_k has spectral gap $\lambda_2(\Delta_k) \geq \gamma\hat{\lambda}$, there exists a classical algorithm*

Algorithm	**Complexes**	**Complexity is** $\operatorname{poly}(n)$ **if**
quantum algorithm of [LGZ16]	general	$\gamma, \varepsilon \in \Omega(1/\operatorname{poly}(n))$
this work	general	$\gamma, \varepsilon \in \Omega(1)$
this work	clique, $k \in \Omega(n)$	$\gamma \in \Omega(1), \varepsilon \in \Omega(1/\operatorname{poly}(n))$ or $\gamma \in \Omega(1/\log^2(n)), \varepsilon \in \Omega(1)$

Table 3.1 – Comparison of the parameter settings of quantum and classical algorithms for the Betti number estimation problem under which their running time is polynomial.

that, for any $\varepsilon > 0$, outputs with high probability an estimate $\tilde{\nu}_k = \beta_k/d_k \pm \varepsilon$ of the k-th (normalised) Betti number of a general simplicial complex in time

$$n^{\mathcal{O}\left(\frac{1}{\sqrt{\gamma}} \log \frac{1}{\varepsilon}\right)},$$

and of a clique complex in time

$$\left(\frac{n}{\hat{\lambda}}\right)^{\mathcal{O}\left(\frac{1}{\sqrt{\gamma}} \log \frac{1}{\varepsilon}\right)} \cdot \operatorname{poly}(n).$$

The algorithm has space complexity $\operatorname{poly}(n, 1/\gamma, \log(1/\varepsilon))$.

For general simplicial complexes, our algorithm improves upon the aforementioned classical algorithms if $k \in \Omega(1/\sqrt{\gamma})$. Now let us focus on the special case of clique complexes. Since $n \geq \lambda_{\max}(\Delta_k) \geq k + \delta_k + 1$ (see Lemma 3.2.4), with δ_k being the maximum up-degree over all k-faces (see Definition 3.2.1), we can simply set $\hat{\lambda} = n$ if $k \in \Omega(n)$ or if we know that $\delta_k \in \Omega(n)$. In such case, the algorithm for clique complexes runs in time $2^{\mathcal{O}\left(\frac{1}{\sqrt{\gamma}} \log \frac{1}{\varepsilon}\right)} \cdot \operatorname{poly}(n)$. This is polynomial if either $\gamma \in \Omega(1)$ and $\varepsilon = 1/\operatorname{poly}(n)$, or $\gamma \in \Omega(1/\log^2 n)$ and $\varepsilon \in \Omega(1)$. The algorithm provides a classical counterpart to the aforementioned line of quantum algorithms for estimating Betti numbers which, under similar assumptions, have a runtime scaling as $\operatorname{poly}(n, 1/\gamma, 1/\varepsilon)$. We summarize these findings in Table 3.1.

The complexity of our algorithm for general simplicial complexes can alternatively be obtained using the singular value transformation (SVT) algorithm by Gharibian and Le Gall, or more precisely, the "*dequantized quantum singular value transformation algorithm*" in Section 4 of [GLG22]. The main difference is that we use a path integral Monte Carlo approach for computing matrix powers, instead of computing them explicitly as in [GLG22, Lemma 3]. This approach provides us with an exponential improvement in the space complexity since the SVT algorithm has space complexity $n^{\mathcal{O}\left(\frac{1}{\gamma} \log \frac{1}{\varepsilon}\right)}$. The more significant benefit is that we get an improved algorithm for clique complexes, which is the main case of interest for the aforementioned quantum algorithms. We show that the k-th combinatorial Laplacian is n-sparse for clique complexes, as compared to general simplicial complexes which are $\mathcal{O}(kn)$-sparse. This implies that it is closer to a diagonally dominant matrix, and we can exploit this for obtaining better time complexity when using the path integral Monte Carlo technique.

3.3.1 Algorithm for general simplicial complexes

Consider a general simplicial complex K with vertex set $[n]$, d_k denoting the number of its k-faces, Δ_k its k-th combinatorial Laplacian, and with k-th Betti number β_k (over a field of characteristic 0, like $\mathbb{Q}$ or $\mathbb{R}$) . We wish to obtain an estimate $\tilde{\nu}_k$ that satisfies $\tilde{\nu}_k = \beta_k/d_k \pm \varepsilon$ for some parameter $\varepsilon \in (0,1)$. In this section, we make the following assumptions:

1. In time polynomial in n, (a) we can check whether a set is in the complex, and (b) we can sample a k-face from the simplicial complex K uniformly at random[3].
2. We have estimates $\hat{\lambda}$ and γ on the largest eigenvalue and the spectral gap of Δ_k, respectively, satisfying

$$\lambda_{\max}(\Delta_k) \leq \hat{\lambda} \leq c\lambda_{\max}(\Delta_k) \quad \text{and} \quad \lambda_2(\Delta_k) \geq \gamma\hat{\lambda},$$

for some constant $c > 0$. If we do not have such a bound on the spectral gap, an alternative is to approximate the "quasi-Betti number", which is the number of small eigenvalues (below $\gamma\hat{\lambda}$) of the combinatorial Laplacian, as in [CC24].

Note that instead of the first assumption we could say that we have query (and sampling) access to the complex where we can ask for any subset of the vertex set whether they are in the complex. This way learning the whole complex would take an exponential number of queries (in n), so it is interesting to have subexponential query complexity. In the following we continue to focus on time, because efficient time complexity is a stronger result than query complexity.

We could introduce the normalised Laplacian $\Delta_k/\lambda_{\max}$ which has its spectrum between 0 and 1. Instead, we consider the related matrix

$$H = I - \Delta_k/\hat{\lambda},$$

which, as we discuss below, satisfies $0 \preceq H \preceq I$. From Lemma 3.2.4, we know that $0 \leq (\Delta_k)_{ii} \leq n$ and Δ_k has $\mathcal{O}(nk)$ nonzero off-diagonal entries in every row, each of absolute magnitude 1. This implies that

$$\|H\|_1 = \max_j \sum_i |H_{ij}| \in \mathcal{O}(nk).$$

By construction, the k-th combinatorial Laplacian Δ_k is positive semidefinite, hence all eigenvalues are non-negative, β_k of them are equal to 0, the second smallest distinct eigenvalue is $\lambda_2(\Delta_k)$, and the maximum eigenvalue is $\lambda_{\max}(\Delta_k)$. Thus, by linearity, the eigenvalues of H lie between 0 and 1, β_k of them equal to 1, and all other eigenvalues lie below $1 - \lambda_2(\Delta_k)/\hat{\lambda} \leq 1 - \gamma$. The following lemma shows how to relate the trace of H^r, for sufficiently large r, to the Betti number β_k.

Lemma 3.3.2. *If $r \geq \frac{1}{\gamma}\log\frac{1}{\varepsilon}$ then $\beta_k \leq \mathrm{Tr}\,(H^r) \leq \beta_k + \varepsilon d_k$.*

Proof. On the one hand, we have that

$$\mathrm{Tr}\,(H^r) = \sum_{i=1}^{d_k} \lambda_i(H)^r \geq \beta_k.$$

3. Assumption (b) is automatically satisfied if the complex is dense in k-faces.

On the other hand, we have that

$$\mathrm{Tr}\left(H^r\right) = \sum_{i=1}^{d_k} \lambda_i(H)^r \leq \beta_k + \sum_{i:\lambda_i(H)<1} (1-\gamma)^r \leq \beta_k + \varepsilon d_k,$$

where we used that $(1-\gamma)^r \leq \varepsilon$ for $r \geq \frac{1}{\gamma}\log\frac{1}{\varepsilon}$. □

Using this observation, we can obtain a 2ε-additive estimate of β_k/d_k from an ε-additive estimate of $\mathrm{Tr}\left(H^r\right)/d_k$. To obtain the latter we use another observation, that holds not only for large r as above, but for a general nonnegative z-th power of H.

Usually we have e.g. $i \in [d_k]$ and $S_i \in F_k$, that is S_i is a k-face and i is its id. In the following, for simplicity but with a slight abuse of notation, i is also going to denote the k-face itself.

Observation 3.3.3.

$$\frac{1}{d_k}\mathrm{Tr}\left(H^z\right) = \frac{1}{d_k}\sum_{i=1}^{d_k} \langle i|H^z|i\rangle = \mathop{\mathbb{E}}_i\left[X_z^{(i)}\right],$$

where $i \in [d_k]$ is sampled uniformly at random and $X_z^{(i)} = \langle i|H^z|i\rangle$.

Since H is $\mathcal{O}(nk) \in \mathcal{O}(n^2)$-sparse, we can evaluate $X_z^{(i)}$ exactly in time $\mathcal{O}(n^{2z})$. Indeed, this is the approach in [GLG22]. Here we use another approach based on the path integral Monte Carlo method, which has two advantages. First, it improves the space complexity from $n^{\mathcal{O}(z)}$, as in [GLG22], to $\widetilde{\mathcal{O}}(nz)$. Second, it will lead to a faster algorithm for clique complexes (see next section).

Let us denote the sign of $H_{i,j}$ by $(-1)^{s(i,j)}$, with $s(i,j) \in \{0,1\}$. We can rewrite $X_z^{(i)}$ as follows:

$$\begin{aligned} X_z^{(i)} &= \langle i|H^z|i\rangle \\ &= \sum_{j_1,\ldots,j_z} \langle i|j_z\rangle \langle j_z|H|j_{z-1}\rangle \ldots \langle j_1|H|i\rangle \\ &= \sum_{j_1,\ldots,j_z} Y_z(i,j_1,\ldots,j_z)\frac{|H_{j_z,j_{z-1}}|}{\|H_{\cdot,j_{z-1}}\|_1}\cdots\frac{|H_{j_1,i}|}{\|H_{\cdot,i}\|_1}, \end{aligned}$$

with

$$Y_z(j_0,j_1,\ldots,j_z) = \langle j_0|j_z\rangle \prod_{\ell=0}^{z-1}(-1)^{s(j_{\ell+1},j_\ell)}\|H_{\cdot,j_\ell}\|_1.$$

By Lemma 3.2.4, it holds that $|Y_z| \leq \|H\|_1^z \leq (n+nk)^z \in \mathcal{O}(n^{2z})$. By our choice of normalisation, we can interpret $|H_{j,i}|/\|H_{\cdot,i}\|_1 =: P(i,j)$ as a transition probability from face i to face j. We can then say that

$$X_z^{(i)} = \mathop{\mathbb{E}}_{(j_0=i,j_1,\ldots,j_z)}\left[Y_z(j_0,j_1,\ldots,j_z)\right],$$

where the path $(j_0, j_1, \ldots, j_z)$ is drawn with probability $P(j_0, j_1) \ldots P(j_{z-1}, j_z)$ from the resulting Markov chain with transition matrix P. Moreover, if we choose the initial k-face $j_0 \in [d_k]$ uniformly at random, then

$$\begin{aligned}\mathbb{E}\left[Y_z\right] &= \mathop{\mathbb{E}}_{(j_0,j_1,\ldots,j_z)} \left[Y_z(j_0, j_1, \ldots, j_z)\right] \\ &= \mathop{\mathbb{E}}_{j_0}\left[X_z^{(j_0)}\right] = \frac{1}{d_k}\operatorname{Tr}(H^z).\end{aligned}$$

This gives us an unbiased estimator Y_z for the normalised trace of H^z. Moreover, as proven in the following lemma, we can sample Y_z efficiently.

Lemma 3.3.4. *We can sample from Y_z, as defined above, in time $z \cdot \mathrm{poly}(n)$.*

Proof. We can evaluate Y_z by sampling z steps of the Markov chain over k-faces. The initial k-face j_0 is drawn uniformly at random. By our assumptions, we can do this in time $\mathrm{poly}(n)$. Subsequent steps are sampled as follows.

Let j_i be the current k-face. First, we learn the up-degree $d^{up}_{j_i}$, and hence $(\Delta_k)_{j_i j_i}$. We do this by, for all potential up-neighbours (obtained by adding one element to the face), querying whether they are in the complex. This takes $n - k - 1$ queries, and hence time $\mathrm{poly}(n)$. Then we learn all down-up neighbours by querying all $\mathcal{O}(n^2)$ subsets with symmetric difference 2. This again takes time $\mathrm{poly}(n)$. By Lemma 3.2.3, we can now derive all $\mathcal{O}(n^2)$ nonzero entries of the j_i-th row $H_{\cdot,j_i}$, and hence sample j_{i+1} according to the probability $P(j_{i+1}, j_i) = |H_{j_{i+1},j_i}|/\|H_{\cdot,j_i}\|_1$. This yields time $\mathrm{poly}(n)$ per step of the Markov chain, and so $z \cdot \mathrm{poly}(n)$ time overall. □

Intuitively, the algorithm does the following. Starting from a random k-face S we do a random walk where in each step we move to a k-face S' that is a down-up neighbour of S but not an up-down neighbour of it (see Lemma 3.2.3). If, after some number of steps, we get back to the starting face S, it means that with high probability, there is a hole of k-faces.

It remains to bound the complexity of estimating $\mathbb{E}[Y_z]$, given samples of Y_z. For this, we use Hoeffding's inequality (Lemma 2.4.1), which yields the following lemma.

Lemma 3.3.5. *For any $\delta > 0$ and integer $z \geq 0$, we can obtain a δ-additive estimate of $\mathbb{E}[Y_z] = \operatorname{Tr}(H^z)/d_k$ by taking the average of $\mathcal{O}(n^{4z})/\delta^2$ many independent samples of Y_z.*

Proof. We know that $|Y_z| \leq \|H\|_1^z \leq (n + nk)^z \in \mathcal{O}(n^{2z})$ (Lemma 3.2.4). Consider p independent samples $Y_{z,1}, \ldots, Y_{z,p}$ distributed according to Y_z. For any $\delta > 0$, Hoeffding's inequality (Lemma 2.4.1) states that

$$\Pr\left(\left|\frac{1}{p}\sum_{i=1}^{p} Y_{z,i} - \mathbb{E}[Y_z]\right| \geq \delta\right) \leq 2\exp\left(\frac{-2p\delta^2}{\mathcal{O}(n^{4z})}\right).$$

If we choose $p = \mathcal{O}(n^{4z})/\delta^2$ then $\frac{1}{p}\sum Y_{z,i}$ will be δ-close to its expectation $\mathbb{E}\left[Y_z\right] = \frac{1}{d_k}\operatorname{Tr}(H^z)$ with probability at least $1 - 1/2^{\mathrm{poly}(n)}$. □

This leads to Algorithm 1, which has time complexity $\mathcal{O}(n^{4z}/\delta^2)$.

For $r \geq \frac{1}{\gamma}\log\frac{2}{\varepsilon}$, we know from Lemma 3.3.2 that $\operatorname{Tr}(H^r)/d_k = \beta_k/d_k \pm \varepsilon/2$. Hence, setting $\delta = \varepsilon/2$ and $z = r$ in the algorithm above we get an ε-additive estimate of β_k/d_k. The algorithm requires $p = \mathcal{O}(n^{4r}/\delta^2) = n^{\mathcal{O}\left(\frac{1}{\gamma}\log\frac{1}{\varepsilon}\right)}$ samples of Y_r, each of which can be obtained in time $r \cdot \mathrm{poly}(n)$ by Lemma 3.3.4. The overall time complexity of Algorithm 1 is hence $n^{\mathcal{O}\left(\frac{1}{\gamma}\log\frac{1}{\varepsilon}\right)}$.

Algorithm 1: Algorithm for δ-estimating $\mathrm{Tr}\left(H^z\right)/d_k = \mathrm{Tr}\left(\left(I - \Delta_k/\hat{\lambda}\right)^z\right)/d_k$

Input: Query and sample access to complex K, integer k, parameters $\hat{\lambda}$ and z, precision parameter $\delta \in (0,1)$.
Output: Estimate $\mathrm{est}_{k,z}$ such that $\mathrm{est}_{k,z} = \mathrm{Tr}(H^z)/d_k \pm \delta$ with high probability.

```
1 Set $p = \mathcal{O}(n^{4z})/\delta^2$.
2 for $t = 1, \dots, p$ do
3     Sample a $k$-face $j_0$ of $K$ uniformly at random.
4     Sample $z$ steps $(j_0, j_1, \dots, j_z)$ of the Markov chain $P$ with initial face $j_0$.
5     Set $Y_{z,t} = \langle j_0 | j_z \rangle \prod_{q=0}^{z-1} (-1)^{s(j_{q+1}, j_q)} \|H_{\cdot, j_q}\|_1$.
6 Return $\mathrm{est}_{k,z} = \frac{1}{p}\sum_{t=1}^{p} Y_{z,t}$.
```

Improvement using Chebyshev polynomials

We can slightly improve this result by approximating H^r with a polynomial of degree roughly $\sqrt{r}$ using Chebyshev polynomials, and then estimating the monomials using Algorithm 1. Let $T_i(x)$ denote the i-th Chebyshev polynomial (of the first kind). They are defined by recurrence $T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$, with $T_0(x) = 1$, $T_1(x) = x$. Chebyshev polynomials are useful in approximation theory, for example in the following lemma.

Lemma 3.3.6 (Follows from [SV14, Theorem 3.3]). *For any $\delta > 0$ and $d \geq \sqrt{2r\log(2/\delta)}$, the monomial x^r can be approximated by a polynomial $p_{r,d}(x)$ of degree d such that $|p_{r,d}(x) - x^r| \leq \delta$ for all $x \in [-1,1]$.*

Now we bound the size of the monomial coefficients in $p_{r,d}(x)$, as these will govern the precision with which we need to estimate the trace of each monomial. Following [SV14, Chapter 3], the polynomial $p_{r,d}(x)$ is obtained by first approximating x^r in the Chebyshev basis by

$$p_{r,d}(x) = \alpha_0^{(r)} + \sum_{i=1}^{d} 2\alpha_i^{(r)} T_i(x)$$

where $\alpha_i^{(r)} = \binom{r}{(r-i)/2}/2^r$ if i has the same parity as r, and $\alpha_i^{(r)} = 0$ otherwise. We then obtain the desired coefficients of $p_{r,d}$ in the monomial basis by expressing each of the Chebyshev polynomials in the monomial basis. Concretely, if $T_i(x) = \sum_{\ell=0}^{i} c_\ell^{(i)} x^\ell$ then

$$\begin{aligned} p_{r,d}(x) &= \alpha_0^{(r)} + \sum_{i=1}^{d} 2\alpha_i^{(r)} T_i(x) \\ &= \alpha_0^{(r)} + \sum_{\ell=0}^{d}\left[\sum_{i=\ell}^{d} 2\alpha_i^{(r)} c_\ell^{(i)}\right] x^\ell =: \sum_{\ell=0}^{d} b_\ell^{(r,d)} x^\ell. \end{aligned}$$

To bound the coefficients $b_\ell^{(r,d)}$, we first bound the coefficients $c_\ell^{(i)}$ in Lemma 3.3.7 below. Combined with the bounds $\left|\alpha_i^{(r)}\right| \leq 1$, this lemma yields the upper bound $|b_\ell| \leq (d+1) \cdot 2 \cdot 2^{2d} \leq 2^{3d}$.

Lemma 3.3.7. *For all $i \in \mathbb{N}$ and $\ell \leq i$, we have $\left|c_\ell^{(i)}\right| \leq 2^{2i}$.*

Proof. We prove the lemma using induction on i. Remember that the Chebyshev polynomials can be defined via the following recurrence

$$T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x),$$

with $T_0(x) = 1$, $T_1(x) = x$. This immediately shows that the bounds $\left|c_\ell^{(i)}\right| \leq 2^{2i}$ hold for $i = 0$ and $i = 1$. Now assume $i > 1$ and that for all $i' < i$ and $\ell \leq i'$, we have $\left|c_\ell^{(i')}\right| \leq 2^{2i'}$. Then from the recursion $T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$, we obtain $c_\ell^{(i)} = 2c_{\ell-1}^{(i-1)} - c_\ell^{(i-2)}$ and hence $\left|c_\ell^{(i)}\right| \leq 2^{2(i-1)+1} + 2^{2(i-2)} \leq 2 \cdot 2^{2(i-1)+1} = 2^{2i}$. □

Now we can describe an efficient algorithm that, for any $\varepsilon > 0$, outputs an additive ε-estimate of β_k/d_k with high probability. The correctness and complexity of the algorithm are proven in Theorem 3.3.8.

Algorithm 2: Algorithm for ε-estimating β_k/d_k

```
Input: Query and sample access to complex $K$, integer $k$, estimates of $\hat{\lambda}$ and $\gamma$,
       precision parameter $\varepsilon \in (0,1)$.
Output: Estimate $\tilde{\nu}_k$ such that $\tilde{\nu}_k = \beta_k/d_k \pm \varepsilon$ with high probability.
1 Set $r = \left\lceil \frac{1}{\gamma} \log \frac{3}{\varepsilon} \right\rceil$ and $d = \left\lceil \sqrt{\frac{2}{\gamma}} \log \frac{6}{\varepsilon} \right\rceil$.
2 for $\ell = 0, \ldots, d$ do
3     Estimate $\mathrm{Tr}(H^\ell)/d_k$ to additive precision $\delta = \varepsilon / \left(3(d+1)2^{3d}\right)$ with high
      probability using Algorithm 1. Let $\mathrm{est}_{k,\ell}$ denote the output.
4 Return $\tilde{\nu}_k = \sum_{\ell=0}^{d} b_\ell^{(r,d)} \mathrm{est}_{k,\ell}$.
```

Theorem 3.3.8. *Algorithm 2 returns with high probability an estimate of β_k/d_k with additive error ε in time $n^{\mathcal{O}\left(\frac{1}{\sqrt{\gamma}} \log \frac{1}{\varepsilon}\right)}$.*

Proof. First, we prove the correctness. By our choice of r, we know from Lemma 3.3.2 that $\mathrm{Tr}(H^r)/d_k = \beta_k/d_k \pm \varepsilon/3$, so it suffices to return an $(2\varepsilon/3)$-additive estimate of $\mathrm{Tr}(H^r)/d_k$. By Lemma 3.3.6, we can use the approximation

$$\frac{1}{d_k} \mathrm{Tr}(H^r) = \frac{1}{d_k} \sum_{\ell=0}^{d} b_\ell^{(r,d)} \mathrm{Tr}\left(H^\ell\right) \pm \varepsilon/3$$

for $d = \left\lceil \sqrt{2r \log \frac{6}{\varepsilon}} \right\rceil \leq \left\lceil \sqrt{\frac{2}{\gamma}} \log \frac{6}{\varepsilon} \right\rceil$. We estimate each term $\mathrm{Tr}\left(H^\ell\right)/d_k$ to precision $\delta = \frac{\varepsilon}{3(d+1)2^{3d}}$ with high probability, so that the final estimator has a total error

$$\begin{aligned} \tilde{\nu}_k = \sum_{\ell=0}^{d} b_\ell^{(r,d)} \mathrm{est}_{k,\ell} &= \sum_{\ell=0}^{d} b_\ell^{(r,d)} \left(\mathrm{Tr}\left(H^\ell\right)/d_k \pm \delta\right) \\ &= \left(\frac{1}{d_k} \sum_{\ell=0}^{d} b_\ell^{(r,d)} \mathrm{Tr}\left(H^\ell\right)\right) \pm \varepsilon/3, \end{aligned}$$

using that $\left|\sum_{\ell=0}^{d} b_\ell^{(r,d)}\delta\right| \leq \delta(d+1)2^{3d} \leq \varepsilon/3$. Combined with the previous error bounds, this shows that $\tilde{\nu}_k = \beta_k/d_k \pm \varepsilon$ with high probability.

To bound the time complexity, recall that the time complexity of Algorithm 1 in Line 3 is $\mathcal{O}(n^{4\ell}/\delta^2) \in n^{\mathcal{O}(d)}/\varepsilon^2$. Summing over the $d+1$ loops, and using the expression for d, this yields a total time complexity that is $n^{\mathcal{O}\left(\frac{1}{\sqrt{\gamma}}\log\frac{1}{\varepsilon}\right)}$. □

This completes the proof of the first item of Theorem 3.3.1.

3.3.2 Algorithm for clique complexes

The complexity of our path integral Monte Carlo algorithm is dominated by the sample complexity that follows from Hoeffding's inequality (Lemma 2.4.1), which we bound using the fact that $|Y_z| \leq \|H\|_1^z$ and $\|H\|_1 = \mathrm{poly}(n)$. Here we prove a tighter bound on $\|H\|_1$ for the special case of clique complexes and exploit this to improve the algorithm.

We will use the following characterisation of the off-diagonal elements of the combinatorial Laplacian Δ_k.

Lemma 3.3.9 (Follows from Lemma 3.2.3)**.** *Let Δ_k denote the k-th combinatorial Laplacian of a simplicial complex K. Then $(\Delta_k)_{ij}$ for $i \neq j$ is nonzero if and only if the corresponding two k-faces are down-up neighbours but not up-down neighbours.*

The following claim is going to be useful for the proof of the next lemma.

Claim 3.3.10. *In a clique complex, every k-face has at most $n-k-1$ down-up neighbours that are not its up-down neighbours.*

Proof. Since we are in the clique complex case, a k-face is exactly a $(k+1)$-clique, so we will use the two expressions interchangeably. We will prove a slightly stronger statement: every vertex that is not in a $(k+1)$-clique C can appear in at most one down-up neighbour of C that is not its up-down neighbour. For contradiction, let us suppose that there is a vertex v among the $n-k-1$ vertices that are not in C such that v belongs to two distinct down-up neighbours C_1 and C_2 of C. That is, suppose there are two distinct vertices $u_1, u_2 \in C$ such that $C_1 = C \setminus \{u_1\} \cup \{v\}$ and $C_2 = C \setminus \{u_2\} \cup \{v\}$ are $(k+1)$-cliques. We show that C_1 and C_2 are up-down neighbours of C. Indeed, v must be adjacent to every vertex of C: from $C_1 \in K$ it is adjacent to all vertices in C other than the vertex u_1, and from $C_2 \in K$ it is adjacent all vertices except for u_2. So $C \cup \{v\}$ forms a $(k+2)$-clique and hence C_1 and C_2 are up-down neighbours of C. □

This section's main observation is the following.

Lemma 3.3.11. *The k-th combinatorial Laplacian of a clique complex has at most $n-k-d_i^{up}$ nonzero entries in every row, that is,*

$$|\{j : (\Delta_k)_{ij} \neq 0\}| \leq n-k-d_i^{up} \quad \forall i \in d_k$$

where d_i^{up} is the up-degree of the k-face corresponding to the i-th row of the combinatorial Laplacian.

Proof. Because of Claim 3.3.10, every k-face has at most $n-k-1$ down-up neighbours that are not its up-down neighbours. Following Lemma 3.3.9, these elements correspond exactly to the nonzero off-diagonal entries in Δ_k. Adding the diagonal element $(\Delta_k)_{ii}$, we obtain that the total number of nonzero entries in a row of the k-th combinatorial Laplacian is at most $n-k$.

To improve this bound, notice that if a vertex v is adjacent to all the vertices of a k-face (i.e, we have an up-neighbouring $(k+1)$-face), then v cannot be in any down-up neighbour that is not an up-down neighbour as well. Thus, using Lemma 3.3.9 again, we can say that every up-neighbour "cancels" the corresponding down-up neighbours in Δ_k.

Hence, if the k-face corresponding to the i-th row of the combinatorial Laplacian has up-degree d_i^{up}, then the number of nonzero entries in the i-th row of Δ_k is not more than $n-k-d_i^{\text{up}}$. □

From this, we get the following corollary.

Corollary 3.3.12. $\|H\|_1 \leq 2n/\hat{\lambda}$.

Proof. Recall that $H = I - \Delta_k/\hat{\lambda}$. Thus,

$$\begin{aligned}\|H\|_1 &= \max_j \sum_i |H_{ij}| \\ &\leq \max_j \left|(I)_{jj} - \frac{(\Delta_k)_{jj}}{\hat{\lambda}}\right| + n \cdot \frac{1}{\hat{\lambda}} \leq 2\frac{n}{\hat{\lambda}},\end{aligned}$$

where for the first inequality, we used the fact that $|(\Delta_k)_{ij}|$ is either 1 or 0 if $i \neq j$, and by Lemma 3.3.9, it is 1 at most n times in every row or column. In the second inequality, we used the fact that $0 \leq (\Delta_k)_{ii} \leq n$. Combining, we have the result. □

Now let us recall the path integral estimator Y_z as defined in the previous section:

$$Y_z(j_0, j_1, \dots, j_z) = \langle j_0 | j_z \rangle \prod_{\ell=0}^{z-1} (-1)^{s(j_{\ell+1}, j_\ell)} \|H_{\cdot, j_\ell}\|_1.$$

As a consequence of Corollary 3.3.12, it satisfies

$$|Y_z| \leq \|H\|_1^z \leq \left(\frac{2n}{\hat{\lambda}}\right)^z.$$

This improves the sample complexity in Lemma 3.3.5 from $\mathcal{O}\left(n^{4z}\right)/\delta^2$ to $\mathcal{O}\left(\left(\frac{2n}{\hat{\lambda}}\right)^{2z}\right) \cdot \frac{1}{\delta^2}$ that is $\left(\frac{n}{\hat{\lambda}}\right)^{\mathcal{O}(z)} \cdot \frac{1}{\delta^2}$. This directly propagates to Algorithm 2, improving its time complexity from

$$n^{\mathcal{O}\left(\frac{1}{\sqrt{\gamma}} \log \frac{1}{\varepsilon}\right)} \text{ to } \left(\frac{n}{\hat{\lambda}}\right)^{\mathcal{O}\left(\frac{1}{\sqrt{\gamma}} \log \frac{1}{\varepsilon}\right)} \cdot \mathrm{poly}(n).$$

The $\mathrm{poly}(n)$ term comes from the time required for sampling (Lemma 3.3.4). This completes the proof of the second item of Theorem 3.3.1.

3.4 Property testing very large Betti numbers

As a reminder, in graph property testing we wish to decide whether a graph has a certain property, or whether it is "far" from having that property [Gol10]. In the *dense* graph model, an n-vertex graph is ε-far from having a property if we have to add or remove more than εn^2 edges for the graph to have the property. A *tester* for a given property is a randomised algorithm that, given query access to the adjacency matrix of a graph G, can distinguish with constant success probability whether G has that property or is ε-far from having it. A graph property is said to be *testable* if there exists a tester that makes a number of queries that is a function only of ε, and so independent of the graph size. Examples of testable properties are bipartiteness, triangle-freeness and, more generally, monotone (closed under removing edges) and hereditary (closed under removing vertices) graph properties [AS05, AS08].

In this section, we prove the following theorem (for a more formal statement see Theorem 3.4.7).

Theorem 3.4.1 (Informal)**.** *The property of a clique complex having a (very) large k-th Betti number is testable for constant k.*

Recall that on an intuitive level, the k-th Betti number β_k of a clique complex defined by underlying graph G counts the number of independent k-dimensional "holes" in the complex, which is bounded by the number of $(k+1)$-cliques d_k in G. More formally, β_k equals the rank of the k-th *homology group*.

In order to prove the theorem, in Section 3.4.1 we use the matroid notion of independence to relate the Betti number β_k to the number of "independent" K_{k+2} cliques in the graph, and we bound the total number of cliques as a function of the number of independent cliques. Then, in Section 3.4.2, we build on these tools to reduce the problem of testing large Betti numbers to that of (tolerantly) testing clique-freeness, which is known to be testable. In particular, we show that for any constant k, having a large Betti number implies that the graph is close to being K_{k+2}-free, while being far from having a large Betti number implies that the graph is far from being K_{k+2}-free.

We prove the result for

$$\delta(\varepsilon,0)=\sqrt{2\varepsilon}\ , \qquad \delta(\varepsilon,1)=\varepsilon/3\ , \qquad \delta(\varepsilon,k)=1/\mathrm{tower}(k^4\log(1/\varepsilon))\ \ (k>1).$$

Here the $\mathrm{tower}(\ell)$-function denotes a height-ℓ tower of powers of 2's – this explains the extra quantifier in "*(very)* large Betti numbers". Nonetheless, this property is neither trivial for constant k and ε nor monotone or hereditary. To see this, consider the $(k+1)$-partite graph which has $d_k=(\frac{n}{k+1})^{k+1}$ and $\beta_k=(\frac{n}{k+1}-1)^{k+1}$ (Proposition 3.2.5), and so $\beta_k/d_k=1-O(k^2/n)$. This shows that for any k and $\delta>0$, there exist graphs with the property $\beta_k\geq(1-\delta)d_k$. (Compare this to our Proposition 3.4.9 which states that any large clique complex that has few k-faces is close to having a large k-th Betti number.) Moreover, the quantity β_k/d_k increases as a function of n, so the property cannot be monotone or hereditary.

3.4.1 Betti numbers via independent faces

In this section, we connect the number of independent k-faces with the total number of k-faces, and connect the Betti number β_k to the number of independent k- and $(k+1)$-faces in the complex.

The notion of independence of faces in Definition 3.2.10 leads to the following useful observation. It is a direct consequence of the fact that a set of k-faces has zero boundary if and only if the sum of the corresponding boundary vectors is the zero vector.

Proposition 3.4.2. *In a k-dimensional simplicial complex (i.e., $|F_{k+1}| = 0$), a set of k-faces is independent iff no subset of them forms a k-dimensional hole.*

The independence of holes is defined similarly. A k-dimensional hole is a set of k-faces (an element of C_k), and associated to it is a characteristic vector over $\{0,1\}^{d_k}$. Remember that in Section 3.2.3 we associated the same kind of (boundary) vectors to $(k+1)$-faces: in this sense a k-dimensional hole can be seen as the boundary of a virtual $(k+1)$-dimensional object. This way, a set of holes is independent if the corresponding vectors are linearly independent (over field $\{0,1\}$). An analogue of Proposition 3.4.2 tells us that a set of k-dimensional holes is independent iff no subset of them (as virtual $(k+1)$-faces) forms a $(k+1)$-dimensional hole.

Let us denote the rank $\mathrm{rk}(M_k(K))$ of the k-simplicial matroid, i.e. the size of a maximal independent set of k-faces in K, by $r_k(K)$. Alternatively, we can say that $r_k = \dim(\mathrm{im}(\partial_k))$. We are going to need a lower bound on this value in terms of the total number of k-faces $d_k(K)$. For the sake of completeness, we also include an upper bound in the statement.

Lemma 3.4.3. *For any $0 \leq k < n$ and any simplicial complex K*

$$\frac{k+1}{n} d_k(K) \leq r_k(K) \leq \min\left\{ d_k(K), \binom{n-1}{k} \right\}.$$

Proof. Trivially, $r_k(K) \leq d_k(K)$. Moreover, the set of k-faces $F_k(K)$ of any complex K can be obtained from that of the full complex $F_k(K_k^{\text{full}})$ by removing faces, and this can only decrease the rank. Combined with Proposition 3.2.11, we get $r_k(K) \leq \binom{n-1}{k}$.

Now let us prove the main part of the claim, which is the lower bound. We use a similar argument to the one below Proposition 3.2.11. Let u be a vertex in K that is included in a maximum number of k-faces (i.e., the vertex with the highest "k-face-degree"). These k-faces that contain u are independent because each contains a $(k-1)$-face that the others do not (the one without u), and this is a non-zero element in their boundary vector. As there are d_k many k-faces in K, each incident to $k+1$ vertices, the average "k-face-degree" of a vertex is $(k+1)d_k/n$. Thus, the independent set of k-faces defined by u has at least this many k-faces, and so $r_k \geq (k+1)d_k/n$. □

The next lemma shows a nice connection between the rank, the number of faces and the Betti number. For $k = 0$, the formula gives the well-known graph formula $c = n - t$, where t is the number of edges in a spanning forest, n is the number of vertices, and c is the number of connected components. When $k = 1$ and the underlying graph is connected and planar, it gives the Euler formula $n + f = e + 2$ (with n the number of vertices, f the number of faces surrounded by edges and e the number of edges) because $\beta_1 = f - 1$, $d_1 = e$, $r_1 = n - 1$ and $r_2 = 0$.

Lemma 3.4.4 (e.g. [Nan] Proposition 3.13.)**.** *For any simplicial complex,*

$$\beta_k = d_k - r_k - r_{k+1}.$$

Proof. By applying the rank–nullity theorem to ∂_k, we get $d_k = \dim\ker\partial_k + \dim\operatorname{im}\partial_k$. Now notice that $\dim\operatorname{im}\partial_k = r_k$ because the independence of k-faces is defined through their boundary vectors (Definition 3.2.10), thus we have $\dim\ker\partial_k = d_k - r_k$. From Definition 2.6.5 we can see that $\beta_k = \dim\ker\partial_k - \dim\operatorname{im}\partial_{k+1}$. Substituting what we got before, we obtain $\beta_k = (d_k - r_k) - r_{k+1}$. □

We also give an alternative, combinatorial proof of this lemma, which ties closer to the spirit of this work.

Proof. The proof goes by induction. Let Δ denote the simplicial complex being considered and let us take a basis of the k-simplicial matroid over Δ. For the base case, we consider the subcomplex where this is the set of all k-faces and all the higher dimensional faces are removed, in which case $r_k = d_k$ and $\beta_k = r_{k+1} = 0$ and so the formula holds. In the inductive step, we will put back all the removed faces. We start by adding the rest of the k-faces one by one, and we argue that each added face creates exactly one new independent hole.

First, note that adding a dependent k-face S to the complex creates at least one hole (otherwise we could have added it to the basis by Proposition 3.4.2). Moreover, the hole is independent of the previous ones because it contains the face S, which no other hole contains so far.

Then, we prove that adding a k-face creates at most one hole. For contradiction, assume that there is a k-face S such that when added to the set, more than one new independent holes are created. We consider two of them, $\{S, R_1, \ldots, R_p\}$ and $\{S, T_1, \ldots, T_q\}$, which we call the "R-hole" and the "T-hole". Necessarily, they have zero boundary (we denote the boundary vectors the same way as the k-faces):

$$S + R_1 + \cdots + R_p = 0$$
$$S + T_1 + \cdots + T_q = 0.$$

Adding the equations shows that $\{R_1, \ldots, R_p, T_1, \ldots, T_q\}$ must also be a hole, call it the "RT-hole". It does not contain S, so it must have been present before adding S. However, by construction, the R-, T- and RT-holes are not independent, and so we get a contradiction.

Let Δ_k denote the complex we have now: it contains exactly the faces of Δ up to dimension k, and no faces of higher dimension. So far we proved that $r_k = d_k - \beta_k(\Delta_k)$. Let us consider the set of "fillable" k-cycles in Δ_k, i.e. sets H of cardinality $k+2$ where all the $(k+1)$-subsets of H are in Δ_k. These are those holes of Δ_k that may be filled by $(k+1)$-faces in Δ.

Now we continue the induction by adding to Δ_k the $(k+1)$-faces of Δ one by one (and in the end the higher dimensional faces as well) to get back Δ. Each $(k+1)$-face fills a fillable cycle, and it is independent of the previously added ones if and only if the hole being filled is independent of the previously filled ones (as they are the same subset). Thus, every time Δ gains an independent $(k+1)$-face it loses an independent k-hole. This finishes the proof, as adding faces of dimension larger than $k+1$ does not change any parameter in the claim. □

In the special case where $k = 0$ and the graph defined by the vertices and edges of K is connected, we have $\beta_0 = 1$, $d_0 = n$ and $r_{k+1} = n - 1$ (a spanning tree of the graph is a maximal independent edge set). Thus, r_0 has to be defined as 0, which makes sense if we think about r_k as $r_k = \dim(\operatorname{im}(\partial_k))$.

Remark 3.4.5. *Let $T_k = \{S \subseteq V, |S| = k+1\}$ and $A \subseteq T_k$. In early works [CR70, Cor78], only complexes of the form $K_{k-1}^{\text{full}} \cup A$ are analysed in detail. This family of complexes is not enough to express the k-th Betti number of an arbitrary simplicial complex. For example, for this restricted class of complexes Cordovil [Cor78, Proposition 1.2] showed that $r_k = d_k - \beta_k$, which is only a special case of Lemma 3.4.4 (with $r_{k+1} = 0$).*

An easy consequence of Lemma 3.4.4 is the following statement.

Proposition 3.4.6. *For any simplicial complex K and $k \geq 1$, $\beta_k(K) \leq \binom{n-1}{k+1}$.*

Proof. In K_k^{full}, we have $\beta_k = d_k - r_k - 0 = \binom{n}{k+1} - \binom{n-1}{k} = \binom{n-1}{k+1}$, and removing k-faces or adding $(k+1)$-faces cannot increase this value. $\square$

3.4.2 Testing large Betti numbers

Now we turn to proving the main result of this section: that we can test whether a Betti number is large. Below, we state our main theorem formally.

Theorem 3.4.7 (formal version of Theorem 3.4.1)**.** *Consider a clique complex K given by query access to its underlying graph in the dense graph model. For any constant k and $\varepsilon > 0$, there exists $\delta(\varepsilon, k) > 0$ such that the graph property of having k-th Betti number $\beta_k(K) \geq (1-\delta)d_k$ (over $\mathbb{F}_2$) is testable for any $\delta \leq \delta(\varepsilon, k)$ (with distance parameter ε).*

Even though our results in Section 3.4.1 hold for general simplicial complexes, the main theorem is restricted to clique complexes. The reason for this is that we wish to phrase our results in the well-established setting of graph property testing. By constraining ourselves to clique complexes, having a large Betti number becomes a graph property (of the underlying graph) rather than a property of an abstract simplicial complex. Also, this way we can use some previous results from graph property testing, like the tolerant testability of subgraph freeness (Lemma 3.2.7).

Warm-up: testing many components

The 0-th Betti number β_0 of a clique complex K equals the number of connected components of the underlying graph G. As an informal warm-up and a blueprint for the general case, we show how to test whether $\beta_0 \geq (1-\delta)n$. The argument involves two reductions.

First, we argue that having a large 0-th Betti number is equivalent to having few independent edges. From Lemma 3.4.4, we get that $\beta_0 = n - r_1 - r_0$ where $r_0 = 0$. Thus, $\beta_0 \geq (1-\delta)n$ is equivalent to $r_1 \leq \delta n$, so testing large β_0 reduces to testing whether G has a small number of independent edges.

Now comes the second reduction, in which we argue that testing whether G has few independent edges can be reduced to *tolerantly* testing edge-freeness. For this, note that if G has $r_1 \leq \delta n$ independent edges then the total number of edges $|E| \leq \binom{\delta n+1}{2} < \delta^2 n^2/2 + O(n)$. Note that if we applied Lemma 3.4.3, we would get $|E| \leq r_1 n/2 \leq \delta n^2/2$. We get the better bound by noticing that if there are δn independent edges, then we have a maximum number of edges if all the independent edges are in the same connected component and this component is a $K_{\delta n+1}$.

$|E| < \delta^2 n^2/2 + O(n)$ implies that so G is $1.1\delta^2/2$-close to being edge-free. On the other hand, if G is ε-far from having $r_1 \leq \delta n$, then G must also be ε-far from having $r_1 = 0$, i.e from being edge-free. Therefore, we reduced the problem of testing $\beta_0 \geq (1-\delta)n$ to that of tolerantly testing edge-freeness (with parameters $\varepsilon_1 = 1.1\delta^2/2$ and $\varepsilon_2 = \varepsilon$). It remains to note that edge-freeness is tolerantly testable by Lemma 3.2.7.

General case

We now turn to proving our general result (Theorem 3.4.7), that having a Betti number $\beta_k \geq (1-\delta)d_k$ is testable for constant k. Following the blueprint from the $k=0$ case, we first reduce the problem to testing whether there are few independent $(k+1)$-faces, and then reduce testing few independent $(k+1)$-faces to tolerantly testing $(k+2)$-clique freeness.

We consider a clique complex K with underlying graph G. From Lemma 3.4.4, we get that

$$d_k - r_{k+1} \geq \beta_k = d_k - r_{k+1} - r_k, \tag{3.1}$$

from which we can prove the following lemma.

Lemma 3.4.8 (Large Betti number $\preceq$ few independent cliques)**.** *In a clique complex K with underlying graph G, if $\beta_k \geq (1-\delta)d_k$ then $r_{k+1} \leq \delta d_k$. If G is ε-far from having $\beta_k \geq (1-\delta)d_k$ in K, then it is $\varepsilon/2$-far from having $r_{k+1} = 0$, i.e., G is $\varepsilon/2$-far from K_{k+2}-freeness.*

Proof. The first part of the claim is clear from Equation (3.1). For the second part (being ε-far), we will use the definition of ε-far (Definition 2.3.1). That is, we want to prove that if every graph that is ε-close to G has $\beta_k < (1-\delta)d_k$ then every graph that is $\varepsilon/2$-close to G satisfies $r_{k+1} > 0$.

For contradiction, assume that there is a particular H that is $\varepsilon/2$-close to G but has $r_{k+1} = 0$. Since H is ε-close to G, it has $\beta_k < (1-\delta)d_k$, or equivalently $r_k + r_{k+1} > \delta d_k$ (using Lemma 3.4.4). Because of this, we have $d_k < r_k/\delta \leq \binom{n-1}{k}/\delta$ (by Lemma 3.4.3). With the construction of Proposition 3.4.9 below, we can modify H to get an H' that is $\alpha = \varepsilon/2$-close to H (thus still ε-close to G) and that has $\beta_k \geq (1-\delta)d_k$. This contradicts the assumption that every graph that is ε-close to G satisfies $\beta_k < (1-\delta)d_k$. □

Proposition 3.4.9. *Consider a graph $H = (V, E)$ with $|V| = n$ sufficiently large, and assume that H has at most $\binom{n-1}{k}/\delta$ k-faces. Then for any constant proximity parameter α, there is another graph H' that is α-close to H and has $\beta_k \geq (1-\delta)d_k$.*

Proof. We give a construction that modifies H to get H'. Let us choose any vertex set $S \subseteq V$ of size $|S| = \alpha n$. We delete all the edges that go between S and $V \setminus S$, and we modify the edges within S to construct a complete $(k+1)$-partite subgraph. This modifies at most αn^2 edges, so yields a graph H' that is α-close to H.

The subgraph of H' induced by S is a complete $(k+1)$-partite graph with $\left(\frac{\alpha n}{k+1}\right)^{k+1}$ many k-faces. Thus, in H' we have at most this amount plus the number of original k-faces of H, i.e. $d_k(H') \leq \left(\frac{\alpha n}{k+1}\right)^{k+1} + \binom{n-1}{k}/\delta$. The number of independent k-holes in the subgraph of H' induced by S is $\left(\frac{\alpha n}{k+1} - 1\right)^{k+1}$ (see Proposition 3.2.5), so in H' it is at least this much: $\beta_k(H') \geq \left(\frac{\alpha n}{k+1} - 1\right)^{k+1}$. Clearly $\beta_k(H')/d_k(H') = 1 - O(1/n)$. For any $\delta > 0$ this is at least $1-\delta$ for n sufficiently large. □

Remark 3.4.10. *In Lemma 3.4.8, the second proximity parameter is not necessarily half of ε, it can be arbitrarily close to it. For example, it could be 0.99ε, but then we have to use the construction of Proposition 3.4.9 with $\alpha = 0.01\varepsilon$ instead of $\varepsilon/2$.*

For our second reduction, we use Lemma 3.4.3, which tells us that if $r_{k+1} \leq \delta d_k$ then

$$d_{k+1} \leq \frac{\delta}{k+2} n d_k \leq \frac{\delta}{k+2} n \binom{n}{k+1} \leq \frac{\delta}{(k+2)!} n^{k+2}.$$

Combined with Lemma 3.4.8, we get that $\beta_k \geq (1-\delta)d_k$ implies $d_{k+1} \leq \frac{\delta}{(k+2)!} n^{k+2}$. We see that a large Betti number implies a small number of K_{k+2}-s in the graph, while being far from having a large Betti number implies being far from K_{k+2}-freeness (by Lemma 3.4.8).

In fact, by the graph removal lemma (Lemma 3.2.6), a small number of K_{k+2}-s implies that the graph is close to being K_{k+2}-free. More specifically, for any $\varepsilon' > 0$ there exists $\delta = \delta(k, \varepsilon') > 0$ such that if G has at most $\frac{\delta}{(k+2)!} n^{k+2}$ many K_{k+2}-s then G is ε'-close to being K_{k+2}-free. By picking (say) $\varepsilon' = \varepsilon/2$, it follows that we can test whether $\beta_k \geq (1-\delta)d_k$ by tolerantly testing whether G is $\varepsilon/2$-close or ε-far from K_{k+2}-freeness. By Lemma 3.2.7, we know that K_{k+2}-freeness is indeed tolerantly testable, and this proves our main Theorem 3.4.7.

To finish, we comment on the complexity of the algorithm. The complexity of (non-tolerant) K_{k+2}-freeness testing is dominated by $1/\delta$. Using Remark 3.2.8, we obtain a complexity of $2^{\mathrm{poly}(1/\varepsilon) \cdot 2^{O(1/\delta)}}$ for the tolerant version, which is what our algorithm uses. Hence, we need to focus on the scaling of $\delta = \delta(k, \varepsilon)$.

The current best upper bound in the graph removal lemma requires $\delta(k, \varepsilon) \leq 1/\mathrm{tower}(5(k+2)^4 \log(1/\varepsilon))$ [Fox11]. As a reminder, $\mathrm{tower}(i)$ is a tower of twos of height i (e.g., $\mathrm{tower}(3) = 2^{2^2}$). For the case of $k = 0$, we could avoid this: recall from Section 3.4.2 that $r_1 \leq \delta n$ implies that G is $\delta^2/2$-close to being edge-free. Similarly, for $k = 1$ we can get a better bound: $r_2 \leq \delta n^2$ implies that G is 3δ-close to being triangle-free. Indeed, if we remove all the edges of a maximal independent triangle set (at most $3\delta n^2$ edges), then any remaining triangle in the graph would contradict the maximality of the chosen set. We leave the extension of similar arguments to higher k for future work. In conclusion, we get a tester that distinguishes $\beta_k \geq (1-\delta)d_k$ from being ε-far under the constraints

$$\delta < \sqrt{2\varepsilon} \ \ (k=0), \quad \delta < \varepsilon/3 \ \ (k=1), \quad \delta < 1/\mathrm{tower}(5(k+2)^4 \log(1/\varepsilon)) \ \ (k>1).$$

Chapter 4

Quantum property testing

4.1 Introduction

Some researchers have already considered efficient quantum algorithms for property testing both classical and quantum objects, see for instance [BFNR08, ABRW16, HLM17, BDCG+20, AS19] and the survey [MdW16]. Notably, the authors in [ACL11] initiated the study of bounded degree graph property testing in the quantum model. One important result in this context is the result of [BDCG+20], who proved that there can be exponential quantum advantage in the bounded degree graph model of property testing. However, as mentioned in their paper, the graph property admitting the exponential quantum advantage is not a natural one.

4.1.1 Property testing of directed bounded degree graphs

While all the aforementioned works consider undirected graphs, many real-world instances (such as the world wide web) correspond to *directed* graphs. Consequently, Bender and Ron [BR02] introduced a model of property testing for directed graphs in the classical setting, focusing on the properties of acyclicity and connectivity. Following that work, we open a new research line by studying quantum algorithms for testing directed graphs. As we will see, by doing so we address new fundamental questions in the field of quantum query complexity. Answering them requires using recent techniques and partially answering some new or open questions.

As described in [BR02], for bounded-degree directed graphs there are two natural query models: (i) the *unidirectional* model, where the algorithm is allowed to query the outgoing edges of a vertex, but not the incoming edges, and (ii) the *bidirectional* model, where the algorithm can query both the incoming and outgoing edges of a vertex. Interestingly, [BR02] showed that strong connectivity is testable in the bidirectional model (i.e., it can be tested with a number of queries that depends on ε but not on N), but it requires $\Omega(\sqrt{N})$ queries in the unidirectional model. Later, the testability of other graph properties like Eulerianity, vertex and edge connectivity [OR11, YI10b, FNY+20, CY19] was also shown in the bidirectional model. While there is a clear distinction between the two models, Czumaj, Peng and Sohler [CPS16] showed that if a property is testable in the bidirectional model, then it has a *sublinear* (i.e., $o(N)$) query complexity in the unidirectional model.

In this chapter, we consider a particularly important problem in the unidirectional model: the problem of testing *subgraph-freeness*. More precisely, we examine the problem of property testing "*k-source-subgraph-freeness*", where the goal is to test H-freeness for some constant-sized subgraph H with k "source components", where a source component is a strongly connected subgraph that has no incoming edges. This problem was first studied by Hellweg and Sohler [HS12], and they provided a testing algorithm that performs $O(N^{1-1/k})$ queries. They also proved a tight lower bound of $\Omega(N^{2/3})$ for the $k = 3$ case (see [HS12, Theorem 1 and Theorem 3]).

Recently, Peng and Wang [PW23] proved a matching lower bound for any constant k. In particular, they showed that $\Omega(N^{1-\frac{1}{k}})$ queries are necessary for testing k-star-freeness (which is a special case of k-source-subgraph-freeness) in the unidirectional model, for arbitrary k (see [PW23, Theorem 1.2]). Notice that asymptotically the complexity of testing k-star-freeness becomes $\Omega(N)$. This also proves that the aforementioned reduction of [CPS16] cannot be made much stronger: for the property of k-star-freeness, the separation between the query complexities in the bi- and unidirectional models is maximal, because this property can be tested using constant number of queries in the bidirectional model.

4.1.2 Related works on collision finding

A closely related problem to finding k-stars in graphs is finding k-collisions in integer sequences. The two mentioned classical papers on subgraph-freeness testing [HS12, PW23] actually consider a collision-type intermediate problem for proving their lower bounds. As we are also going to do so, let us look at some related, known results.

The problem of *collision finding* is a ubiquitous problem in the field of algorithm theory with wide applications in cryptography. Here, given a sequence s of N integers, the goal is to find a duplicate in s. If one has the guarantee that $\Theta(N)$ elements of the sequence are duplicated, which is the case, for instance, when the sequence consists of uniformly random integers from $[N]$, it is well-known that classically $\Theta(\sqrt{N})$ queries are necessary and sufficient due to the birthday paradox. In the quantum model, this can be solved with query complexity $\Theta(N^{1/3})$ by the algorithm of Brassard, Høyer and Tapp [BHT98]. The matching lower bound was first stated for a specific set of hard instances known as 2-to-1 (i.e., each integer appears exactly twice or not at all) by Aaronson and Shi [AS04]. For some constant integer $k \geq 3$, those results can be further extended to finding k-collisions in a random input with suitable alphabet size, so that it contains $\Theta(N)$ k-duplicates with high probability. The classical query complexity for this problem is $\Theta(N^{1-1/k})$ [HS12, PW23], and quantumly it is $\Theta\left(N^{\frac{1}{2}\left(1-\frac{1}{2^k-1}\right)}\right)$ [LZ19]. The situation is more complex for non-random inputs.

Remarkably, the complexity of *testing* k-collision-freeness (i.e., the absence of k-collisions) is harder to settle on the lower bound side than the *finding* version. In this work, we are going to focus on the hardness of distinguishing inputs that have linearly many collisions from those that do not have any. For $k = 2$, the two problems have the same complexity, since intuitively the only way to distinguish is to find a collision. This can be formalised easily in the classical case. Quantumly, this is more challenging, but the lower bound in [AS04] proved the hardness of distinguishing between 2-to-1 instances and ones with no duplicate.

However, for larger k, distinguishing such inputs might be easier than finding a collision. The classical upper bound of $O(N^{1-1/k})$ queries is straightforward for the finding variant. In the lower bounds of [HS12, PW23], the authors consider the distinguishing version, so classically the question is settled. But in the quantum setting, the upper and lower bounds of [LZ19] are tight only for finding k-collisions in random inputs, and for the distinguishing variant, we are not currently aware of anything better than the $\Omega(N^{1/3})$ lower bound corresponding to the $k = 2$ case. To our knowledge, prior to our work, this problem has not yet been studied in the quantum setting.

4.1.3 Our results

In this chapter, based on our paper **[AMSS25]**, we present two lines of results for quantum property testing of graph properties.

In the first line, we consider the problem of testing k-source-subgraph-freeness in the unidirectional model. This problem is almost maximally hard for large k in the classical regime, and we show that it admits an almost quadratic advantage in the quantum setting.

Theorem 4.1.1 (Restated in Theorem 4.3.3)**.** *The quantum query complexity of testing k-source-subgraph-freeness in the unidirectional model is $O\left(N^{\frac{1}{2}\left(1-\frac{1}{2^k-1}\right)}\right)$.*

In order to prove the above result, we connect it to the problem of finding k-collisions. In [LZ19], an algorithm is given for finding k-collisions in sequences of random integers. We generalise this to the context of graph property testing in two ways: first, finding a subgraph (instead of a collision); and second, considering graphs that are far from being H-free (instead of random).

Moreover, we prove that this quantum advantage is nearly tight, by showing a quantum lower bound using the method of dual polynomials.

Theorem 4.1.2 (Corollary of Theorem 4.1.3)**.** *The quantum query complexity of testing k-source-subgraph-freeness in the unidirectional model is $\tilde{\Omega}\left(N^{\frac{1}{2}\left(1-\frac{1}{k}\right)}\right)$.*

For proving graph property testing lower bounds, both the classical works of [HS12] and [PW23] prove collision testing lower bounds using the proportional moments technique of [RRSS09]. At the heart of this technique is a construction of two positive integer random variables, X_1 and X_2, with different expectations but with the following conditions on the first $k-1$ moments: $\mathbb{E}[X_1]/\mathbb{E}[X_2] = \mathbb{E}[X_1^2]/\mathbb{E}[X_2^2] = \ldots = \mathbb{E}[X_1^{k-1}]/\mathbb{E}[X_2^{k-1}]$. Such a construction leads to a randomised query complexity lower bound of $\Omega(N^{1-\frac{1}{k}})$ for various property testing problems such as k-collision-freeness [PW23]. Having a quantum version of this technique has been identified as an important open problem [ABRW16], since this could be used to pave the way to stronger quantum lower bounds in related settings. We modestly made progress to this quest for the special case of testing k-collision-freeness.

In [LZ19], in addition to the algorithm we mentioned, they also prove a matching lower bound showing that their algorithm for *finding* k-collisions in random inputs is optimal. However, this time we cannot reuse those techniques for our purpose for two main reasons. First, the property testing variant of this problem could be easier. Moreover, their lower bound technique requires random inputs and hence it does not apply to our case. This is why we use yet another method, that of dual polynomials, to prove our lower bound.

Theorem 4.1.3 (Restated in Theorem 4.4.1)**.** *The quantum query complexity of testing k-collision-freeness is $\tilde{\Omega}\left(N^{\frac{1}{2}\left(1-\frac{1}{k}\right)}\right)$.*

In the second line of results, we show that not all problems in graph property testing admit such a quantum speedup. This fact even remains valid for the case of undirected graphs both in the bounded-degree and dense models. In the bounded-degree model, we consider the property testing variant of the famous problem of 3-colourability: namely, distinguishing whether an unknown undirected graph G on N vertices can be properly coloured with 3 colours, or one needs to modify a large fraction of its edges to make it 3-colourable. In the classical bounded degree setting, this problem has been studied by [BOT02], who proved a lower bound of $\Omega(N)$ queries. In this work, we present a simple argument that proves that there exists no sublinear quantum tester either for this problem. Our result is stated as follows:

Theorem 4.1.4 (Restated in Theorem 4.5.1)**.** *The quantum query complexity of property testing 3-colourability in undirected bounded-degree graphs is $\Omega(N)$.*

We complement this with a similar result in the dense graph model: there is a graph property that, when testing an N-vertex graph, has asymptotically maximal $\Omega(N^2)$ query complexity, even for quantum algorithms. Here, the property considered is not a natural one, it was described in [GKNR12, Appendix A] for proving the lower bound in the classical setting. We adapt their proof to prove the following statement.

Theorem 4.1.5 (Restated in Theorem 4.6.1)**.** *There is a graph property in the dense model that its property testing requires quantum query complexity $\Omega(N^2)$.*

4.1.4 Technical overview

Subgraph-finding algorithm

We start by describing how to prove the upper bound result of Theorem 4.1.1 for testing k-source-subgraph-freeness. We view the problem as a generalisation of the problem of finding k-collisions and adapt an existing quantum algorithm for the latter problem. In [LZ19], an algorithm is given for finding k-collisions in length-N sequences of integers that contain $\Omega(N)$ k-collisions (e.g. k-to-1, or random sequence with appropriate parameters). Their algorithm generalises the well-known collision finding algorithm of [BHT98]. On a high level, the [LZ19] algorithm first finds several 2-collisions using Grover search like in [BHT98], extends some of them to 3-collisions in a similar way, and so on until a k-collision is found.

On the one hand, instead of random inputs, we consider the problem in the property testing context; and on the other hand, we generalise collision-finding to subgraph-finding. As a first step let us look at what happens when we consider the property testing version of the k-collision problem. In order to be able to use the algorithm of [LZ19], we have to prove that if a length-N sequence is far from k-collision-freeness then it contains many k-collisions. Notice that the collisions are not necessarily distinct: if the input only contains the same integer N times, it only contains one huge collision, but it is still ε-far from k-collision-freeness for any $\varepsilon < 1-k/N$. Thus, what we need to show is that there are $\Omega(N)$ many disjoint size-k sets of indices such that for each set, the sequence contains the same character at the positions of the set. This statement is true because otherwise, by modifying all the characters that are in positions contained in a set ($o(N)$ characters in total), we could get a k-collision-free sequence which contradicts being far from k-collision-freeness.

When we make the second step of turning to testing subgraph-freeness, we need to prove a variant of this statement: if an N-vertex graph G is far from H-freeness (for some constant-sized subgraph H) then it contains $\Omega(N)$ many "source-disjoint" H-subgraphs. This means that there are $\Omega(N)$ many such H-subgraphs in G that the set of vertices in the source components of each H-subgraph are disjoint. We prove this fact in Proposition 4.3.2 and this allows us to further generalise the approach of [LZ19]: first find several partial solutions where only a few source components of an H-subgraph are explored, and gradually extend these (using Grover search coupled with constant-depth breadth first search) until a complete H-subgraph is found.

Notice that this way our algorithm *finds* an H-subgraph in G promised that G is far from H-freeness. This task is at least as difficult as property testing, where the algorithm only has to *distinguish* whether G is H-free or far from any H-free graph. So, our algorithm provides an upper bound on the property testing variant of H-freeness.

Collision-freeness lower bound

Now we will discuss our approach to proving the lower bounds of collision-freeness (Theorem 4.1.2) and k-source-subgraph-freeness (Theorem 4.1.3). We first give a simple reduction from k-collision-freeness to k-star-freeness, which is a special case of k-source-subgraph-freeness. This way, it is enough to prove a lower bound on testing k-collision-freeness, and it implies the same result on testing k-source-subgraph-freeness. Since our lower bound approach crucially depends on the (dual) polynomial method, let us start by briefly discussing it.

The (dual) polynomial method As we discussed in Section 2.5.4, the polynomial method is a common way to prove quantum query complexity lower bounds. As a reminder, it relies on the fact that the acceptance probability of a T-query bounded-error quantum algorithm is a polynomial of degree at most $2T$ [BBC$^+$01]. This way, for proving a quantum query complexity lower bound on calculating a function f, it suffices to argue that any approximating polynomial of f has large degree. One of the key properties that such lower bounds exploit is the symmetry that the function f may exhibit, such as invariance under some permutation of the input. For example, the first tight lower bound of $\Omega(n^{1/3})$ for the collision problem was proved in this way [AS04].

The polynomial method can be written in the form of a linear program, of which one can take the dual. By weak LP-duality, when using this dual characterisation for proving a lower bound on function f, one needs to provide a "witness" of the approximating polynomial's high degree, say Δ. This witness is called the *dual polynomial* ψ and, in the easiest case of total Boolean [1] functions $f : \{-1, 1\}^n \to \{-1, 1\}$, it needs to satisfy three properties.

(i) High correlation with f: $\sum_x f(x)\psi(x) > \delta$.

(ii) Normalisation: $\sum_x |\psi(x)| = 1$.

(iii) Pure high degree Δ: $\sum_x p(x)\psi(x) = 0$, for every polynomial p with degree $< \Delta$.

1. In this chapter, we use $\{-1, 1\}$ where -1 corresponds to the "true value".

Above, the summations are all over $x \in \{-1,1\}^n$.

When the function f is partial, i.e. only defined on a subset $D \subset \{-1,1\}^n$, there is some subtlety that could be handled in two ways (or even in a mixture or both): zero-out the dual polynomial outside D (corresponding to “unbounded degree”); or rewrite condition (i) accordingly (corresponding to “bounded degree”):

(i') High correlation with f: $\sum_{x\in D} f(x)\psi(x) - \sum_{x\notin D} |\psi(x)| > \delta$.

Collision function The paper of [BKT20] also used the dual polynomial method for proving quantum lower bounds for many problems, most of them being open before that work. Similarly to that paper, we need to take several steps to be able to use the dual polynomial method for the problem of property testing k-collision-freeness. This problem was not addressed in [BKT20].

One of the main conceptual ideas in [BKT20] is to re-formulate the problem we study as a composition of two simple Boolean functions. In that paper, powerful techniques are also developed in order to design dual polynomials for simple functions that can be composed. A common way of composing dual polynomials (called dual block composition) dates back to [SZ09, Lee09, She13], but [BKT20] provides new tools for handling it efficiently. We are going to reuse some of them, and also extend one in a way.

The first step is to find the right problem that can fit in the framework. We introduce a partial symmetric function F defined on input strings $s = (s_1, \dots, s_N) \in [R]^N$. The domain of F corresponds to the following promise: either F has no k-collision, or it has many k-collisions occurring for distinct values. More formally,

$$F(s) = \begin{cases} -1 & \text{if no integer occurs at least } k \text{ times in } s, \\ 1 & \text{if more than } \gamma R \text{ distinct integers occur at least } k \text{ times in } s, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

This partial function is not a property testing problem, however it corresponds to a special case of testing k-collision-freeness, which is therefore enough to prove lower bounds.

Binary encoding Now we encode the input string $s = (s_1, \dots, s_N) \in [R]^N$ into binary variables $x_{i,j}$ storing whether $s_i = j$, as in [Aar02]. Doing so, starting from the function F above, we end up with a function f defined over binary variables satisfying several symmetries, under the permutation of either i or j in $x_{i,j}$.

Moreover, the symmetries of f allow the extension of the initial function f from the very restricted set of binary inputs corresponding to valid strings, to the more general set of binary inputs of Hamming weight N [ABRW16]. With further technicalities one can also extend f to all binary inputs of Hamming weight *at most* N [BT20]. This is fundamental because instead of being forced to zero out the dual polynomial outside the domain of f, we only need to do so on inputs of Hamming weight higher than N. Using the symmetry of f, it can be shown that a lower bound on this modified, Boolean version implies a lower bound on the original k-collision problem.

This way we end up with two promises on binary-encoded input domain. The first one comes from the function F itself: we have the promise that the input contains either no k-collision or it has many of them at different values. The second promise is the consequence of the encoding: we want the binary encoding to have Hamming weight at most N. Let

D denote the set of binary strings satisfying both promises, and let $H_{\leq N}$ denote the set of binary strings with Hamming weight at most N. For this case, we use the "double-promise" version of the dual polynomial method, where, in order to prove that every δ-approximating polynomial of f has degree at least Δ, the dual polynomial has to satisfy four conditions, where the fourth one corresponds to zeroing out ψ on large Hamming weight inputs [BKT20].

(i') High correlation with f: $\sum_{x\in D} f(x)\psi(x) - \sum_{x\in H_{\leq N}\setminus D} |\psi(x)| > \delta$.

(ii) Normalisation: $\sum_x |\psi(x)| = 1$.

(iii) Pure high degree Δ: $\sum_x p(x)\psi(x) = 0$, for every polynomial p with degree $< \Delta$.

(iv) No support on inputs with large Hamming weight: $\psi(x) = 0$, for every $x \notin H_{\leq N}$.

Composition Coming back now to the definition of our Boolean function f, one can rewrite it as a composition of simpler functions: $\mathrm{GapOR}_R^\gamma \odot \mathrm{THR}_N^k$. Remember that by composition, we mean $(g \odot h)(x) = g(h(x_1), \dots, h(x_n))$ (where $x = (x_1, \dots, x_n)$ and each $x_j = (x_{1,j}, x_{2,j}, \dots, x_{N,j})$ is a binary vector of dimension N) and the domain is restricted to bit strings of Hamming weight at most N. Above, THR_N^k is the threshold function: it is -1 if the input bitstring contains at least k many -1 (true) values, and is 1 otherwise; and GapOR_R^γ is the gap version of OR, which is 1 if the input only consists of 1s, -1 if the input contains at least γR many -1 values, and is undefined otherwise.

In order to give a dual polynomial for this composed function, we start from a dual polynomial for each part of the composition (ϕ and ψ), which were already given in [BKT20] (in different contexts). Then we use a known way [SZ09, Lee09, She13] of composing dual polynomials called the *dual block composition*, which provides a nearly good dual polynomial $\phi \star \psi$ for $\mathrm{GapOR}_R^\gamma \odot \mathrm{THR}_N^k$. Indeed, by construction, the normalisation (ii) and the pure high degree (iii) are guaranteed. However, the issue is that it is not 0 on bitstrings of large Hamming weight thus (iv) is not satisfied, and the high correlation (i') still has to be proved.

To fix (iv), we use another result of [BKT20] which provides another dual polynomial ζ, that is close to $\phi \star \psi$ and that is 0 on inputs having Hamming weight larger than N. Also, it only changes the pure high degree by a polylogarithmic factor. Now the only remaining task is to prove a large enough correlation (i') of $\phi \star \psi$ and $\mathrm{GapOR}_R^\gamma \odot \mathrm{THR}_N^k$, so that ζ still has high enough correlation. This high correlation proof (Lemma 4.4.23) is the most technical part of this chapter.

High correlation: proof of Lemma 4.4.23 The statement we prove is the following high correlation bound:

$$\sum_{x\in D} (\phi \star \psi)(x) \cdot (\mathrm{GapOR}_R^\gamma \odot \mathrm{THR}_N^k)(x) - \sum_{x \notin D} |(\phi \star \psi)(x)| \geq 9/10.$$

In the proof of this lemma, we use Proposition 4.4.21 that is a more general statement of some techniques used in several proofs of [BKT20]. But then we need to diverge from their proof because it crucially relies on a certain one-sided error property (in the sense of [BKT20, Lemma 6.11]) of the inner function of the composition, which is the OR function in their case. Our inner function is the threshold function, which does not satisfy this property, so we have to use some other properties of the dual polynomials in our proof.

This different proof technique, in the more difficult, two-sided error setting, could be a step towards obtaining a more general lower bound technique.

3-colourability lower bound

Let us now discuss our approach to proving the linear lower bound on the quantum query complexity of testing 3-colourability (Theorem 4.1.4). Before proceeding to present our approach, let us briefly discuss the classical lower bound of testing 3-colourability. To prove the classical lower bound of 3-colourability, the authors in [BOT02] first studied another problem called $\mathrm{E}(3, c)$LIN-2, a problem related to deciding the satisfiability of a system of linear equations. More formally, $\mathrm{E}(3, c)$LIN-2 considers a system of linear equations modulo 2, where each equation has 3 variables and every variable appears in at most c equations. Given such a system of linear equations, the goal is to distinguish if it is satisfiable, or at least some suitable fraction of the equations need to be modified to satisfy it. In [BOT02], is was proved that $\Omega(N)$ classical queries to the system of linear equations are necessary for testing $\mathrm{E}(3, c)$LIN-2.

After this, they designed a reduction from $\mathrm{E}(3, c)$LIN-2 to 3-colourability such that satisfying instances of $\mathrm{E}(3, c)$LIN-2 are reduced to 3-colourable graphs, and far from satisfiable instances of $\mathrm{E}(3, c)$LIN-2 are mapped to far from 3-colourable graphs. Combining these two arguments, in [BOT02] it was proved that $\Omega(N)$ classical queries are necessary for testing 3-colourability for bounded degree graphs.

The authors in [BOT02] used Yao's minimax method [Yao77] to prove the linear lower bound in testing $\mathrm{E}(3, c)$LIN-2. In particular, they designed two distributions D_{yes} and D_{no} such that the systems of linear equations in D_{yes} are satisfiable, whereas the systems of linear equations in D_{no} are far from being satisfiable. A crucial ingredient of their lower bound proof is a construction of a system of linear equations (represented as a matrix) that are far from being satisfiable, but any δN rows of the matrix are linearly independent. Hence, any subset of δN entries of the matrix will look uniformly random, and therefore hard to distinguish from a satisfiable instance.

It is a known fact that distinguishing between a uniformly random string and a ℓ-wise independent string is hard for quantum algorithms (see e.g. [ADW22]). Using this result, we can construct suitable hard instances for $\mathrm{E}(3, c)$LIN-2, such that testing $\mathrm{E}(3, c)$LIN-2 remains maximally hard (requires $\Omega(N)$ queries) for any quantum algorithm. Combining this hardness result with the reduction from $\mathrm{E}(3, c)$LIN-2 to 3-colourability, we finally prove that $\Omega(N)$ quantum queries are necessary for testing 3-colourability. We formally prove this in Section 4.5.

Later, in [YI10a], the authors used various reductions to 3-colourability to argue that several other important problems, including testing Hamiltonian Path/Cycle, approximating Independent Set/Vertex Cover size etc., are maximally hard to test in the classical model. As a corollary of our quantum lower bound, we also obtain maximal quantum query complexity for these problems.

A lower bound in the dense graph model

We show that there is a property in the dense graph model, that testing it has essentially maximal, $\Omega(N^2)$ quantum query complexity for N-vertex graphs (Theorem 4.1.5). The property is the same described in [GKNR12, Appendix A]. On a high level, the construction of this property starts by taking a code $\mathcal{C} \subset \{0,1\}^n$ of appropriate size, such that taking a uniformly random $X \in \mathcal{C}$ is ℓ-wise independent for some $\ell \in \Omega(n)$. Codes of this kind exist in the literature, and this property implies that $\Omega(n)$ quantum queries are necessary for distinguishing a uniformly random $X \in \mathcal{C}$ from a uniformly random $Y \in \{0,1\}^n$. In the following, the goal is to turn this into a graph property.

To a bitstring $X \in \{0,1\}^n$ we assign a graph G_1 of N vertices, where $n = \binom{N}{2}$, and X is viewed as the description of the adjacency matrix of G_1. Since we want to obtain a graph property, we will need to ensure invariance under graph isomorphism. But we also want to be able to recover the original bitstring X from the final graph, and for this, we add a gadget to G_1, which results in graph G_2. Now we take a random permutation of the vertices of G_2 to get the final graph G_3. The hard-to-test graph property $\mathcal{P}$ is the set of graphs we can obtain at the end of this construction if we start with a bitstring $X \in \mathcal{C}$.

The proof of this property's hardness uses Yao's principle [Yao77]: we construct two distributions of graphs that are hard to distinguish. In particular, graphs in D_{yes} are obtained by taking a uniformly random $X \in \mathcal{C}$, and putting it into the above construction. D_{no} is defined similarly, but we start with a uniformly random $X \in \{0,1\}^n$. Graphs in D_{yes} always satisfy $\mathcal{P}$, and graphs in D_{no} are far from $\mathcal{P}$ with high probability. If an algorithm could distinguish between D_{yes} and D_{no}, it would also distinguish a uniformly random string from an $\Omega(n)$-wise independent one, but this is known to require $\Omega(n) = \Omega(N^2)$ quantum queries.

4.1.5 Open problems

Our work raises several important open questions. First, there is still a gap between our lower and upper bound on the quantum query complexity of testing k-collision-freeness and k-source-subgraph-freeness. In [MTZ20], the authors keep using the dual polynomial method to improve the lower bound of [BKT20] for the k-distinctness problem. They achieve this by using a slightly different dual polynomial for THR_N^k, where they allow more weight on the false positive inputs. This makes it impossible to prove the high correlation of the dual and the primal function, so they use a modified block composition. Our technique might be combined with this other approach to improve our lower bound to $\tilde{\Omega}(N^{1/2-1/(4k)})$.

The authors in [ABRW16] stated it as an open question if one could use a variant of the proportional moments result of [RRSS09] to prove lower bounds on quantum algorithms. We leave this question open, and conjecture that a similar result holds in the quantum setting with a lower bound of $\Omega\left(N^{\frac{1}{2}\left(1-\frac{1}{k}\right)}\right)$. This work may be considered as a proof of this conjecture for the special case of k-collision-freeness, and we hope that it will serve as a step towards proving it in general.

In [CPS16], it was proved that if a graph property can be tested with $O(1)$ queries in the bidirectional model, then it can be tested using $O(N^{1-\Omega(1)})$ queries in the unidirectional model. It would be very interesting to investigate if it also implies a quantum tester with query complexity say $O(N^{1/2-\Omega(1)})$.

Just like the BHT algorithm for collision finding [BHT98], our subgraph-freeness testing algorithm in Section 4.3 requires QRACM. There is another collision finding quantum algorithm that avoids the QRACM assumption of BHT, but its time complexity is higher: $O(N^{2/5})$ instead of $O(N^{1/3})$ [CNPS17]. We are curious whether a something similar is possible for our problems: still beating classical algorithms while not using much quantum memory.

Connection to distribution testing

For further motivation and open questions, we would like to point out how the problems and models presented in this chapter relate to the property testing of distributions. In distribution testing, we have sampling access to an unknown distribution D and we have to decide if D satisfies a property or it is far from any distribution that satisfies it. Let us consider $D = \{p_1, \dots, p_R\}$ with $\sum_{i\in[R]} p_i = 1$, meaning that a sample from D gives i with probability p_i for all $i \in [R]$. There is a model, where we assume that every p_i is an integer multiple of $1/N$, and sampling access to D is simulated as query access to a string of integers $(s_1, \dots, s_N) \in [R]^N$, where the frequency of each character $i \in [R]$ is $f_i = Np_i$ and the string is randomly permuted. This way, a property of distributions can be translated into a symmetric property of integer strings, which is the same model we had in the case of collision-freeness testing.

A very common task in distribution testing is to decide whether two distributions $D_1 = \{p_1, \dots, p_R\}$ and $D_2 = \{q_1, \dots, q_R\}$ are the same or they are ε-far from each other. The distance measure used is the total variation distance (or statistical difference), i.e. D_1 and D_2 are ε-far if $\sum_{i\in[R]} |p_i - q_i| > \varepsilon$. There are two main settings for this problem.

- Unknown-Unknown (U-U): the algorithm has sampling access to both distributions D_1 and D_2, so this is the case described above.
- Known-Unknown (K-U): the algorithm knows D_1 and it has sampling access to the unknown distribution D_2. As uniform is known to be a maximally hard known distribution, this model is also referred to as uniformity testing, where we want to distinguish $\forall i \in [R]:\ q_i = 1/R$ from $\sum_{i\in[R]} |q_i - 1/R| > \varepsilon$.

As for any property testing problem, we can consider the tolerant version of these problems: distinguish whether D_1 and D_2 are ε_1-close or ε_2-far in total variation distance (for some $\varepsilon_1 < \varepsilon_2$). The following table contains the known results about the classical and quantum query complexity of property testing distributions in the different settings, ignoring the dependence on the proximity parameter(s). We can see that the classical results are all essentially tight.

	Classical		Quantum	
	non-tolerant	tolerant	non-tolerant	tolerant
K-U	$\tilde{\Theta}(\sqrt{N})$	$\Theta(N/\log N)$	$\Theta(N^{1/3})$	$\Theta(\sqrt{N})$
U-U	$\tilde{\Theta}(N^{2/3})$	$\Theta(N/\log N)$	$O(\sqrt{N})$ and $\Omega(N^{1/3})$	$\Theta(\sqrt{N})$

Table 4.1 – Known bounds on the query complexity of distribution testing.

Let us focus on the quantum results. The non-tolerant K-U case for $R = N$ is the same problem as testing collision-freeness: translated to the integer string setting, a uniform distribution corresponds to a string where each integer appears exactly once, so it is a permutation; and a distribution that is ε-far from uniform corresponds to a string where the frequencies satisfy $\sum_{i\in[R]} |f_i - N/R| > \varepsilon N$, i.e. it is ε-far from a permutation. Thus, both the upper and lower bounds follow from those of the collision problem [BHT98, AS04, Kut05].

In [BHH11], the authors give an $O(\sqrt{N})$ algorithm for approximating the statistical distance between two unknown distributions and thus solving the tolerant U-U case, and all the other cases that are special cases of it. In [BKT20] a lower bound of $\Omega(\sqrt{N})$ is given for the problem they call "statistical difference from uniform" that corresponds to the tolerant K-U case, and the lower bound carries over to the more general tolerant U-U case.

An important open question in quantum distribution testing is to give a better algorithm or lower bound in the non-tolerant U-U case. Several of the existing results in distribution testing use models like we do in this chapter. In particular, our lower bound in Section 4.4 is inspired by the mentioned result of [BKT20], and we hope that our results may help future research to resolve this open problem.

4.2 Preliminaries

4.2.1 Notations and basic definitions

When dealing with Boolean variables, we will usually use $b \in \{-1, 1\}$ instead of $b' \in \{0, 1\}$. We can get to one from the other easily with the mapping $b = 1 - 2b'$, or its inverse, which means that -1 is going to be treated as the "true" or "accepting" value. The reason for using $\{-1, 1\}$ is that when dealing with dual polynomials it is easier to use this notation.

We denote by 1^n the length-n binary vector made only of 1s, and respectively -1^n. The Hamming weight $|x|_H$ of $x \in \{-1, 1\}^n$ is then defined as the number of -1s in x, that is $|x|_H = |\{i \in [n] : x_i = -1\}|$. Let $H^n_{\leq w} = \{x \in \{-1, 1\}^n : |x|_H \leq w\}$ denote the set of length-n binary vectors with Hamming weight at most w. For any $x \in \mathbb{R}$, $\operatorname{sgn}(x) = 1$ when $x \geq 0$, and -1 otherwise.

For a polynomial p, let $\deg(p)$ denote its degree. Remember that the composition $f \odot g : \{-1, 1\}^{nm} \to \{-1, 1\}$ of two Boolean functions $f : \{-1, 1\}^n \to \{-1, 1\}$ and $g : \{-1, 1\}^m \to \{-1, 1\}$ is defined as $(f \odot g)(x) = f(g(x_1), \dots, g(x_n))$ where $x = (x_1, \dots, x_n)$ with each $x_i \in \{-1, 1\}^m$.

Finally, throughout this chapter, notations $O(\cdot)$ and $\Omega(\cdot)$ will be hiding the dependencies on parameters ε, k and d that we consider to be constants.

4.2.2 Query complexity on graphs

As a reminder, in the undirected bounded-degree graph model, we have query access to the adjacency list of an undirected graph $G = (V, E)$ with maximum degree d, represented as an oracle $\mathcal{O}_G : V \times [d] \to V \cup \{\perp\}$. For any $v \in V$ and $i \in [d]$, $\mathcal{O}_G(v, i)$ returns the i-th neighbour of v if it exists and $\perp$ otherwise.

For bounded-degree directed graphs, there exist two query models. In the *bidirectional model*, we have access to both the outgoing and incoming edges of each vertex. Correspondingly, it is imposed that both the in- and out-degrees of a vertex are bounded by d. In the *unidirectional model*, we can only make queries to the adjacency list of the outgoing edges, and we impose only that the out-degrees of a vertex are bounded by d. Since in this work the primary focus will be on the latter model, let us formally define it below.

In the unidirectional bounded-degree graph model, we have query access to the adjacency list of a digraph $G = (V, E)$ where the out-degree of every vertex is at most d_{out}: for all $v \in V$: $\deg_{\text{out}}(v) \leq d_{\text{out}}$. This access is represented as an oracle $\mathcal{O}_G^{\text{out}} : V \times [d_{\text{out}}] \to V \cup \{\perp\}$ such that for any $v \in V$ and $i \in [d_{\text{out}}]$, we have the following:

$$\mathcal{O}_G^{\text{out}}(v,i) = \begin{cases} w, & \text{if } w \in V \text{ is the } i\text{-th out-neighbour of } v; \\ \perp, & \deg_{\text{out}}(v) < i. \end{cases}$$

For completeness, we note that in some of the previous work on the unidirectional model they do impose the degree bound on both the out- and in-degree [CPS16]. This is mostly because this makes for an easier comparison between the uni- and bidirectional models, as this way they allow the same set of graphs. In this work we assume that only the out-degrees are bounded by d.

Breadth-first search (BFS) is one of the most fundamental graph algorithms. It explores the input graph $G = (V, E)$ layer by layer: starting from a vertex first it explores its direct neighbours then their neighbours etc. It can be implemented using a queue (FIFO - first in first out) data structure in the following way.

In the beginning, only the starting vertex is in the queue and only it is marked as explored. Then we do the following procedure until the queue is not empty: query and add to the queue each unexplored neighbour of the vertex at the queue head, mark them as explored and remove the vertex head from the queue. The query and time complexity of BFS is $O(|V| + |E|)$.

The vertices can also store their BFS-depth: the depth of a vertex v is 1 plus the depth of the vertex that added v to the queue (and the depth of the starting vertex is 0). This way, it is possible to run BFS up to some limited depth ℓ: vertices that would have depth larger than ℓ are not added to the queue and the algorithm can terminate before exploring the whole graph: it only explores the ℓ-neighbourhood of the starting vertex. If the depth limit ℓ and the maximum degree of G are both constants, then the depth-ℓ BFS algorithm has constant query (and time) complexity.

4.2.3 Problem definitions

We now define the problems we study and argue about certain relations between them. While the problems are phrased as total decision problems, ultimately, we will care about the quantum query complexity for testing the corresponding properties. The complexity is going to be parameterised by a parameter k. Moreover, the parameter k, the degree bound d and the proximity parameter ε, are all considered to be constants throughout this chapter.

Let us start with some definitions that will be useful to define our problems precisely.

Definition 4.2.1 (Source component). *Let $G = (V, E)$ be a digraph. A set $S \subseteq V$ is called a* source component *if it induces a strongly connected subgraph in G, and in G there is no edge from $V \setminus S$ to S.*

Definition 4.2.2 (k-star). *A k-star is a digraph on $k + 1$ vertices and k edges with one centre vertex, and k source vertices connected to the centre vertex.*

We will now state the decision variant of several problems. The "property" corresponding to a decision problem is the set of inputs that should be accepted in the decision problem.

k**-Source-Subgraph-Freeness**

Parameter: Graph H of constant size with at most k source components

Query access: d-bounded out-degree directed graph G on N vertices (unidirectional model)

Task: Accept iff G is H-free, that is, no subgraph of G is isomorphic to H

In [HS12, PW23], the authors examine the classical query complexity of testing k-source-subgraph-freeness. They consider the bounded-degree unidirectional model, albeit with a bound on both the in- and out-degrees.

For proving a lower bound, we will look at a special case of the main problem: k-star-freeness. Notice that a k-star has k source components, hence a lower bound for this problem implies the same lower bound for the more general k-source-subgraph-freeness problem.

k**-Star-Freeness**

Parameter: Integer $k \geq 2$

Query access: d-bounded out-degree directed graph G on N vertices (unidirectional model)

Task: Accept iff G is k-star-free, that is, no subgraph of G is isomorphic to the k-star

For the lower bound on k-star-freeness testing, we are going to use as a "helper problem" the decision variant of the k-collision problem.

k**-Collision-Freeness**

Parameter: Integer $k \geq 2$

Query access: Sequence of integers $s = (s_1, \dots, s_N) \in [R]^N$

Task: Accept iff s is k-collision-free, i.e. there is no $i_1, \dots, i_k \in [N]$ with $s_{i_1} = \cdots = s_{i_k}$

As discussed in the introduction of this chapter (Section 4.1.2), very little was known about the property testing version of this problem prior to this work. We only know the that the complexity is $\Theta(N^{1/3})$ when $k = 2$, and it is between $\Omega(N^{1/3})$ and $O\left(N^{\frac{1}{2}\left(1-\frac{1}{2^k-1}\right)}\right)$ for larger k.

Reduction from k-collision-freeness to k-star-freeness Now we are going to prove that testing k-collision-freeness can be reduced to testing k-star-freeness (or more generally to testing k-source-subgraph-freeness). Thus, a lower bound on testing k-collision-freeness yields a lower bound on testing k-source-subgraph-freeness. Also, an algorithm for testing k-source-subgraph-freeness yields an upper bound on testing k-collision-freeness.

While the proof goes similarly to [HS12, Theorem 3], our reduction is not identical because we have a slightly different "helper problem". Since they consider that the in-degree of vertices to be bounded as well, for the collision problem, they assume that the sequence does not contain any collision of size larger than k (defined as k-*occurrence-freeness*).

Proposition 4.2.3. *The problem of ε-testing k-collision-freeness of a sequence from $[R]^N$ can be reduced to $\frac{\varepsilon N}{d(N+R)}$-testing k-star-freeness of an $(N+R)$-vertex sparse directed graph with out-degree bound $d \geq 1$.*

Proof. Let us assume that we have an algorithm that solves the k-star-freeness testing problem on graphs with out-degree bound $d \geq 1$, and we want to use it to test k-collision-freeness of a sequence $s = (s_1, \dots, s_N) \in [R]^N$. We construct a digraph G that has N outer vertices $u_1, \dots, u_N$ and R inner vertices $v_1, \dots, v_R$; edges only exist from the outer vertices towards the inner ones such that u_i is connected to v_j iff $s_i = j$. Observe that the maximum out-degree in G is 1, so its out-degree is bounded by d for any $d \geq 1$.

It is clear that s is k-collision-free iff G is k-star-free. On the other hand, if s is ε-far from k-collision-freeness, it implies that more than εN edges have to be deleted in G to make it k-star-free. Thus G is $\varepsilon' = \frac{\varepsilon N}{d(N+R)}$-far from k-star-freeness. □

4.3 Quantum algorithm for testing subgraph-freeness

In this section, we prove that there is a quantum speedup for testing H-freeness in directed graphs with d-bounded out-degree, for any graph H that has k source components. For large but constant k, the speedup is nearly quadratic. This problem was studied in [GR02] in the classical setting. Our algorithm can be seen as a generalisation of the one in [LZ19] to graphs and to the property testing setting. Let us start with the definition of source-disjointness which will be used in the analysis of our algorithm.

Definition 4.3.1 (Source-disjointness)**.** *Let G be a directed graph such that it contains two subgraphs H_1 and H_2. We say that H_1 and H_2 are* source-disjoint *if the union of the source components of H_1 is disjoint from the union of the source components of H_2.*

Moreover, we need to prove the following simple proposition. It shows that if G is far from being H-free, then it contains many source-disjoint copies of H, that is, copies of H that are source-disjoint subgraphs of G.

Proposition 4.3.2. *Let H be an h-vertex graph with k source components. Assume that a d-bounded out-degree directed graph G on N vertices is ε-far from H-freeness. Then G contains at least $\varepsilon N/h = \Omega(N)$ source-disjoint copies of H.*

Proof. We prove the result by contraposition. Consider a maximal set M of source-disjoint copies of H in G and assume that $|M| < \varepsilon N/h$. Let U denote the union of all the vertices in the source components of the copies in M. This implies that if one deletes all the outgoing edges of all the vertices in U, then G becomes H-free. Indeed, if there remained an H-copy then all its source components are disjoint from M' (as in a source component every vertex has at least one outgoing edge), contradicting the fact that M was maximal.

Since $|U| \leq |M| \cdot h$, the number of those deleted edges is at most $|U| \cdot d \leq |M| \cdot hd < \varepsilon N d$. Therefore, the resulting graph is both H-free and ε-close to the original graph G. This proves the contraposition of the proposition. □

4.3.1 The algorithm for $k = 2$

To illustrate the algorithm, we first consider the $k = 2$ case to build our intuition. In this case, our algorithm generalises the BHT algorithm for collision finding [BHT98] to graphs. The high-level idea is that if we manage to sample a vertex from each of the two source components of an H-subgraph (a collision) then by querying their "surroundings" we will discover the H-instance. In the following, we set $h = |V(H)|$, number of vertices in H. We use the shorthand BFS for breadth-first search.

1. Sample a uniformly random vertex subset $\mathcal{S}$ of size $t = \Theta(N^{1/3})$ in G. Perform a depth-h BFS from every vertex in $\mathcal{S}$.
2. Perform Grover search over the remaining vertices $V \backslash \mathcal{S}$ in the following way. A vertex v is marked if there exists another vertex $u \in \mathcal{S}$ such that u and v are from the 2 different source components of an H subgraph of G.
3. If any occurrence of H in G is found, output **Reject**. Otherwise, output **Accept**.

Note that if G is H-free, then the above algorithm will always accept. Now we need to argue that if G is ε-far from being H-free, then with constant probability the above algorithm will find a copy of H and thus reject.

By Proposition 4.3.2, with high probability, a constant fraction of the t vertices in $\mathcal{S}$ are part of a source component in source-disjoint H-subgraphs of G. For such vertices, the BFS in step 1 will discover the entire source component, as well as all other vertices reachable from that source component in H. Then, in step 2, we search for a vertex that is in the remaining source component of such an instance of H that we already partly discovered. This can be verified by doing a depth-h BFS from it and checking if this completes an H-instance with one of the previously sampled vertices' neighbourhoods. As we mentioned, by Proposition 4.3.2, with high probability there are $\Omega(t)$ many marked vertices. This proves the correctness of the algorithm.

Finally, we bound the algorithm's query complexity. Step 1 makes $O(t) = O(N^{1/3})$ many (classical) queries. In step 2, we use Theorem 2.5.4 and Remark 2.5.5: checking whether a vertex is marked requires running a depth-h BFS from it, which costs $c = O(1)$ queries. We argued that there are $\Omega(t)$ many marked vertices, so Grover search makes $O(\sqrt{N/t}) = O(N^{1/3})$ quantum queries.

Note that, just like in the original BHT algorithm, we need QRACM (Quantum Random Access Classical Memory) to perform step 2, because Grover search needs to access the results of the classical queries of step 1 in superposition. In particular, the oracle of Grover search, that tells which elements are marked, will be implemented as a QRACM.

4.3.2 The algorithm for general k

We are now ready to state our general upper bound result. The algorithm and proof follow the same lines as the $k = 2$ case.

Theorem 4.3.3 (Restatement of Theorem 4.1.1)**.** *Let H be a digraph of constant size with k source components. The quantum query complexity of testing H-freeness of an N-vertex graph with bounded out-degree in the unidirectional model is* $O\left(N^{\frac{1}{2}\left(1-\frac{1}{2^k-1}\right)}\right)$.

Proof. In order to extend the $k = 2$ case described above to larger k, we first try to find many partial H-instances with $k-1$ source components found, and then extend one of them to a complete H-instance. We present a brief description of our algorithm below, where h is the number of vertices of H:

1. Sample a uniformly random vertex subset $\mathcal{S}_1$ in G of size t_1. Perform a depth-h BFS from every vertex in $\mathcal{S}_1$. Let $\mathcal{S}_1' = \mathcal{S}_1$.

2. For iterations $i = 2$ to $k-1$, do the following:

   (a) Perform Grover search t_i times on the vertices $V \setminus \mathcal{S}'_{i-1}$ in the following way. A vertex v is marked if there exist $i-1$ other vertices $u_j \in \mathcal{S}_j$ for each $j \in [i-1]$ such that $u_1, \dots, u_{i-1}$ and v are from i different source components of an H subgraph of G. If we do not find t_i vertices like this, output **Reject**, otherwise let $\mathcal{S}_i$ denote the set of the vertices v that we found.

   (b) Set $\mathcal{S}'_i = \mathcal{S}'_{i-1} \cup \mathcal{S}_i$.

3. Perform Grover search on $V \setminus \mathcal{S}'_{k-1}$ to find a complete H-instance. I.e., a vertex v is marked if there exist $k-1$ other vertices $u_j \in \mathcal{S}_j$ for each $j \in [k-1]$ such that $u_1, \dots, u_{k-1}$ and v are from the k different source components of an H subgraph of G.

4. If any occurrence of H in G is found, output **Reject** and terminate the algorithm. Otherwise, output **Accept**.

The correctness proof is similar to the $k = 2$ case. Proposition 4.3.2 tells us that in $\mathcal{S}_1$ there are $\Omega(t_1)$ many vertices that are from a source component of an H-copy. Because of the source-disjointness of the H-copies, when $i = 2$, there are $\Omega(t_1)$ many 1-partial solutions that can be extended to a complete H instance by disjoint remaining source components. As Grover search provides uniformly random marked elements, a constant fraction of the t_2 many 2-partial solutions are actually extendable to H in a similar, disjoint way. This continues to be true in each iteration: (with high probability) a constant fraction of the t_{i-1} many $(i-1)$-partial solutions are extendable to complete H instances by disjoint remaining source components. This way, the last step is going to find an H-subgraph with high probability.

To bound the query complexity, first note that in every application of Grover search, checking whether a vertex is marked (depth-h BFS) takes $O(1)$ queries. The first iteration's Grover searches find t_2 partial H-instances with 2 of its source components found, which takes $O(t_2\sqrt{N/t_1})$ queries (by Theorem 2.5.4). Similarly, for i-th iteration there are $\Omega(t_{i-1})$ marked elements (see the argument in the previous paragraph), so the algorithm performs $O(t_i\sqrt{N/t_{i-1}})$ quantum queries for every $i \in [k-1]$. Finally, finding one complete H-instance costs $O(\sqrt{N/t_{k-1}})$ queries. Thus, the total query complexity is $O(t_1 + \sum_{i=1}^{k-1} t_{i+1}\sqrt{N/t_i})$ with $t_k = 1$. Similar to the multi-collision algorithm in [LZ19, Section 3], we can equate all terms by setting $t_i = \Theta\left(N^{\frac{2^{k-i}-1}{2^k-1}}\right)$, which yields the final quantum query complexity $O\left(N^{\frac{1}{2}\left(1-\frac{1}{2^k-1}\right)}\right)$.

We note that there is no need for a polylog(N) factor in the query complexity, which could come from a commonly used way to boost up the success probability of Grover's algorithm. This stems from two observations. First, consider the case where K among N elements are marked, with a given lower bound $L \leq K$, and we wish to find $R \leq L$ such elements. If $R \ll L$, then (say) $100R$ repetitions of Grover should return at least R marked elements with probability at least $2/3$ while making $O(R\sqrt{N/L})$ queries, without extra log-factors. This is because one can simply ignore any unsuccessful Grover runs. In our case we set $R = t_{i+1} \ll L = t_i$. Finally, since there are k iterations in the algorithm and k is constant, a factor of $\log k$ would not add up to the query complexity of our algorithm in terms of N. □

4.4 Collision-freeness lower bound

As discussed in Section 4.2, we are going to prove a lower bound on the problem of testing k-collision-freeness.

Theorem 4.4.1 (Restatement of Theorem 4.1.3)**.** *Let $k \geq 3$ and $0 < \varepsilon < 1/(4^{k-1}\lceil 20(2k)^{k/2}\rceil)$ be constants. Let N be a large enough positive integer. Then the quantum query complexity of property testing of k-collision-freeness of a sequence of integers $S = (s_1, \dots, s_N) \in [N]^N$ with parameter ε is $\Omega(N^{1/2-1/(2k)}/\ln^2 N)$.*

The proof of the theorem is at the end of Section 4.4.3. Observe that Theorem 4.1.2 is implied by Theorem 4.4.1 and the reduction in Proposition 4.2.3. Our proof mostly follows the structure of [BKT20, Section 6.1], and in particular, it uses the notion of dual polynomials for non-Boolean partial symmetric functions. Our main technical contribution in this section is the proof of Lemma 4.4.23, because the corresponding proof in [BKT20] crucially relies on a fact that does not hold for our problem. We will discuss it in detail below.

In the following, we first state some general results related to the polynomial method for non-Boolean functions, then we use these results for our problem to state the exact statement that we prove in the technical part.

4.4.1 The (dual) polynomial method

For Boolean functions We consider a property on Boolean vectors as a function $f : D \subseteq \{-1,1\}^n \to \{-1,1\}$. Since the work of [BBC+01] it has been known that the acceptance probability $p(x)$ of a T-query bounded-error quantum algorithm on input $x \in D$ is a polynomial of degree at most $2T$ (see Section 2.5.4). Thus, since we went from $\{0,1\}$ to $\{-1,1\}$, the polynomial $(1-2p(x))$ must be a good approximation of f. Observe that $(1-2p(x))$ remains bounded outside D since $p(x)$ remains a probability defined by the algorithm, with no constraint.

In order to formalise this, we first define the notion of bounded approximate degree of a partial Boolean function (compare to Definition 2.5.1 for total functions), and then relate it to its query complexity.

Definition 4.4.2 (Bounded approximate degree)**.** *Let $f : D \subseteq \{-1,1\}^n \to \{-1,1\}$ and $\delta > 0$. A polynomial $p : \{-1,1\}^n \to \mathbb{R}$ δ-approximates f on D if*

$$\forall x \in D : \; |f(x) - p(x)| < \delta \quad \textit{and} \quad \forall x \in \{-1,1\}^n \setminus D : \; |p(x)| < 1 + \delta.$$

Moreover, the bounded δ-approximate degree $\mathrm{bdeg}_\delta(f)$ of f on D *is the smallest degree of such a polynomial.*

The following lemma connects quantum query complexity and approximate bounded degree (compare to Theorem 2.5.2 for total functions).

Lemma 4.4.3 ([BBC+01, AAI+16])**.** *Let $f : D \subseteq \{-1,1\}^n \to \{-1,1\}$ and $\delta > 0$. If a quantum algorithm, having query access to any input $x \in D$, computes $f(x)$ with error probability at most δ using T queries, then there is a polynomial of degree at most $2T$ that 2δ-approximates f on D.*

In particular, this implies that the quantum query complexity for computing f with error δ is at least $\mathrm{bdeg}_{2\delta}(f)/2$, and so we will focus on proving lower bounds on the approximate bounded degree.

We now turn to a dual characterization of this polynomial approximation. This method of dual polynomials dates back to [She11, SZ09] for initially studying communication complexity. Below, we refer to some results stated in [BKT20] for studying query complexity.

Definition 4.4.4 (Pure high degree). *A function $\psi : \{-1,1\}^n \to \mathbb{R}$ has pure high degree at least Δ if for every polynomial $p : \{-1,1\}^n \to \mathbb{R}$ with $\deg(p) < \Delta$ it satisfies $\sum_{x\in\{-1,1\}^n} p(x)\psi(x) = 0$. We denote this as $\mathrm{phd}(\psi) \geq \Delta$.*

One can observe that $\mathrm{phd}(\psi) \geq \Delta$ is equivalent to the fact that all the monomials of ψ are of degree at least Δ. Then by weak LP duality, we get the following result.

Theorem 4.4.5. *[BKT20, Proposition 2.3] Let $f : D \subseteq \{-1,1\}^n \to \{-1,1\}$ and $\delta > 0$. Then $\mathrm{bdeg}_\delta(f) \geq \Delta$ iff there exists a function $\psi : \{-1,1\}^n \to \mathbb{R}$ such that*

$$\sum_{x\in D} \psi(x)f(x) - \sum_{x\in\{-1,1\}^n\setminus D} |\psi(x)| > \delta; \tag{4.1}$$

$$\|\psi\|_1 = \sum_{x\in\{-1,1\}^n} |\psi(x)| = 1; \tag{4.2}$$

$$\mathrm{phd}(\psi) \geq \Delta. \tag{4.3}$$

Now we are going to discuss how to extend these results to non-Boolean functions, which is the interesting case for us.

For non-Boolean partial symmetric functions We now consider a property of a sequence of integers as a function $F : D \subseteq [R]_0^N \to \{-1,1\}$. The symbol 0 will play a special role that will be exhibited later. Unfortunately, one cannot just take the polynomial of those integers. The standard approach (see [Aar02]) is to encode $s = (s_1, \ldots, s_N) \in [R]_0^N$ into binary variables $x = (x_{i,j})_{i\in[N], j\in[R]_0} \in \{-1,1\}^{N(R+1)}$ encoding whether $s_i = j$ as follows: $x_{i,j} = -1$ if $s_i = j$, and $x_{i,j} = 1$ otherwise. Let $H_b^{N(R+1)} \subseteq \{-1,1\}^{N(R+1)}$ be the set of all possible encodings of vectors s, that is for every $i \in [N]$ there is exactly one $j \in [R]_0$ such that $x_{i,j} = -1$.

This way, we can represent F as a function $F_b : D_b \to \{-1,1\}$ where $D_b \subseteq H_b^{N(R+1)}$ is the set of valid encodings of D. More precisely, each $x \in D_b$ satisfies two constraints: (1) $x \in H_b^{N(R+1)}$; and (2) x encodes some $s \in D$. Since only inputs $x \in H_b^{N(R+1)}$ correspond to possible input sequences of an algorithm, the polynomials derived from a quantum query algorithm might not be bounded outside of that set. This implies a slight modification on the definition of approximate degree, in order to relate it to query complexity as in [Aar02].

But before doing this, we are going to relax the constraints on the domain D_b in the case of symmetric functions, while we decrease its dimension. When F is *symmetric* (i.e. $F(s) = F(s \circ \pi_N)$ for any permutation π_N of $[N]$), one can instead define a function $F_{\leq N}$ with weaker constraints by removing the variables corresponding to the symbol 0. Define $H_{\leq N}^{NR}$ as the set of length-(NR) binary vectors with Hamming weight at most N. Given any $x \in H_{\leq N}^{NR}$, we define its frequency vector $z(x) = (z_0, z_1, \ldots, z_R)$ with $z_j = \#\{i :$

$x_{ij} = -1\}$, for $1 \leq j \leq R$, and $z_0 = N - z_1 - \ldots - z_R$. From the vector $z(x)$, one can define a valid sequence of integers $s(x) \in [R]_0^N$: it can be any sequence from $[R]_0^N$ that has frequency vector $z(x)$. Now we can define $F_{\leq N}$ on domain $D_{\leq N}$ as

$$D_{\leq N} = \{x \in H_{\leq N}^{NR} : s(x) \in D\} \quad \text{and} \quad F_{\leq N}(x) = F(s(x)).$$

In fact, for the special case of total symmetric functions F, we can transform F_b on $H_b^{N(R+1)}$ to $F_{\leq N}$ on $H_{\leq N}^{NR}$ due to the symmetry of F.

In [Amb05], it was proved implicitly that for symmetric F, both F_b and $F_{\leq N}$ variants are equally hard to approximate by polynomials. We now define the appropriate notion of approximate degree for $F_{\leq N}$ and relate it to the query complexity of F as in [BKT20, Theorem 6.5].

Definition 4.4.6 (Double-promise approximate degree). *Let $F : D \subseteq [R]_0^N \to \{-1,1\}$ be symmetric and $\delta > 0$. Define $H_{\leq N}^{NR} \subseteq \{-1,1\}^{NR}$ and $F_{\leq N} : D_{\leq N} \subseteq H_{\leq N}^{NR} \to \{-1,1\}$ as above. A polynomial $p : \{-1,1\}^{NR} \to \mathbb{R}$* double-promise δ-approximates *F on D if*

$$\forall x \in D_{\leq N} : \ |F_{\leq N}(x) - p(x)| < \delta \quad \textit{and} \quad \forall x \in H_{\leq N}^{NR} \setminus D_{\leq N} : \ |p(x)| < 1 + \delta.$$

Moreover, the double-promise δ-approximate degree $\mathrm{dpdeg}_\delta(F_{\leq N})$ of $F_{\leq N}$ on $D_{\leq N}$ *is the smallest degree of such a polynomial.*

The following lemma connects quantum query complexity and double-promise approximate degree.

Lemma 4.4.7 ([Aar02, Amb05],[BT20, Theorem 3.9]). *Let $F : D \subseteq [R]_0^N \to \{-1,1\}$ be symmetric and $\delta > 0$. Define $H_{\leq N}^{NR} \subseteq \{-1,1\}^{NR}$ and $F_{\leq N} : D_{\leq N} \subseteq H_{\leq N}^{NR} \to \{-1,1\}$ as above. If a quantum algorithm computes F on D with error δ using T queries, then there is a polynomial p of degree at most $2T$ that double-promise 2δ-approximates $F_{\leq N}$ on $D_{\leq N}$.*

As for the Boolean case, this implies that a quantum algorithm computing F with error δ must make at least $\mathrm{dpdeg}_{2\delta}(F_{\leq N})/2$ queries. We can now also take the dual of this characterization.

Theorem 4.4.8 ([BKT20, Proposition 6.6]). *Let $F : D \subseteq [R]_0^N \to \{-1,1\}$ be symmetric. Define $F_{\leq N} : D_{\leq N} \to \{-1,1\}$ as above. Then $\mathrm{dpdeg}_\delta(F_{\leq N}) \geq \Delta$ iff there exists a function $\psi : \{-1,1\}^{NR} \to \mathbb{R}$ such that*

$$\forall x \in \{-1,1\}^{NR} \setminus H_{\leq N}^{NR}, \quad \psi(x) = 0; \tag{4.4}$$

$$\sum_{x \in D_{\leq N}} \psi(x) F^{\leq N}(x) - \sum_{x \in H_{\leq N}^{NR} \setminus D_{\leq N}} |\psi(x)| > \delta; \tag{4.5}$$

$$\|\psi\|_1 = 1 \quad \textit{and} \quad \mathrm{phd}(\psi) \geq \Delta. \tag{4.6}$$

4.4.2 Preparation

Technically, the problem we use in the proof of Theorem 4.4.1 is slightly more restricted than k-collision-freeness: we want to distinguish no k-collision from many distinct collisions of size at least k.

Definition 4.4.9 (Collision function). *Let $\gamma \in (0,1)$. The symmetric function* $\mathrm{Collision}^{k,\gamma}_{N,R} : D_{\mathrm{Collision}^{k,\gamma}_{N,R}} \subset [R]^N \to \{-1,1\}$ *is defined by* $\mathrm{Collision}^{k,\gamma}_{N,R}(s) = -1$ *if no integer occurs at least k times in s,* $\mathrm{Collision}^{k,\gamma}_{N,R}(s) = 1$ *if there are more than γR distinct integers that occur at least k times in s, and it is undefined otherwise.*

Notice that this problem is not a property testing problem, as the outcome is not determined based on the distance between inputs. Nevertheless, it is a valid promise problem and a special case of testing k-collision-freeness, that we use to prove a lower bound on the other problems of interest.

To prove a bound on the Collision function, we will actually relate it to the composition of two more elementary functions $g \odot h$. Let us define (i) the threshold function $\mathrm{THR}^k_N : \{-1,1\}^N \to \{-1,1\}$ which is -1 if the input bitstring contains at least k many -1s, and it is 1 otherwise; and (ii) the gap version of OR, that is $\mathrm{GapOR}^\gamma_R : D_{\mathrm{GapOR}^\gamma_R} \subset \{-1,1\}^R \to \{-1,1\}$ which takes value 1 if the input is 1^R, -1 if the input contains at least γR many -1s, and is undefined otherwise. We show that the double-promise approximate degree of $\mathrm{GapOR}^\gamma_R \odot \mathrm{THR}^k_N$ lower bounds the quantum query complexity of the collision problem.

Lemma 4.4.10. *Let $k \geq 3$, $0 < \gamma < 1$, $\delta > 0$ and $c > 2$ be constants such that $N/c \leq R \leq N/2$. If the double-promise δ-approximate degree of* $\mathrm{GapOR}^\gamma_R \odot \mathrm{THR}^k_N$ *on domain further restricted to $H^{NR}_{\leq N}$ is at least Δ, then every quantum algorithm computing* $\mathrm{Collision}^{k,\gamma/c}_{N,N}$ *with error $\delta/2$ must require at least $\Delta/2$ queries.*

Before proving this lemma, we prove some helper propositions. In order to apply the dual polynomial method for partial symmetric functions, we start by proving that $\mathrm{Collision}^{k,\gamma'}_{N,R'}$ is at least as hard as a very similar problem. We introduce a "dummy-augmented" version $\mathrm{dCollision}^{k,\gamma}_{N,R} : D_{\mathrm{dCollision}^{k,\gamma}_{N,R}} \subseteq [R]^N_0 \to \{-1,1\}$ of the problem $\mathrm{Collision}^{k,\gamma}_{N,R}$ for the purpose of proving Lemma 4.4.10, where now the input sequence can have integer 0, but those 0s are just ignored when they occur. We show that it is enough to prove a lower bound for this second version.

Proposition 4.4.11. *Let $k \geq 3$, $0 < \gamma < 1$ and $c > 2$ be constants such that $N/c \leq R \leq N/2$. Then* $\mathrm{dCollision}^{k,\gamma}_{N,R}$ *can be reduced to* $\mathrm{Collision}^{k,\gamma/c}_{N,N}$.

Proof. An input to $\mathrm{dCollision}^{k,\gamma}_{N,R}$ is a sequence $s = (s_1, \dots, s_N)$ where each $s_i \in [R]_0$. Let us define a family of functions T_i that map from $[R]_0$ to $[R']$ for $R' = R + \lceil N/2 \rceil$: $T_i(s) = s$ if $s > 0$ and $T_i(0) = R + \lceil i/2 \rceil$.

Notice that $(s_1, \dots, s_N)$ is free from k-collisions (ignoring collisions of the dummy character 0) if and only if $(T_1(s_1), \dots, T_N(s_N))$ is free from k-collisions, i.e. new k-collisions cannot be created by this transformation (only 2-collisions but we assume $k \geq 3$).

On the other hand, if $(s_1, \dots, s_N)$ contains more than γR distinct k-collisions, then so does $(T_1(s_1), \dots, T_N(s_N))$. Since $\gamma R \geq (\gamma/c)N$, $\mathrm{Collision}^{k,\gamma/c}_{N,N}$ will reject. □

The following proposition relates dCollision to $\mathrm{GapOR}^\gamma_R \odot \mathrm{THR}^k_N$.

Observation 4.4.12. *The domain of* $\mathrm{GapOR}^\gamma_R \odot \mathrm{THR}^k_N$ *is*

$$D_{\mathrm{GapOR}^\gamma_R \odot \mathrm{THR}^k_N} = \{x \in \{-1,1\}^{NR} : (\mathrm{THR}^k_N(x_1), \dots, \mathrm{THR}^k_N(x_R)) \in H^R_{\geq \gamma R} \cup \{1^R\}\}.$$

where $x = (x_1, \ldots, x_R)$ with each $x_i \in \{-1,1\}^N$.

The domain of $(\mathrm{dCollision}^{k,\gamma}_{N,R})^{\leq N}$ *is*

$$D_{(\mathrm{dCollision}^{k,\gamma}_{N,R})^{\leq N}} = H^{NR}_{\leq N} \cap D_{\mathrm{GapOR}^{\gamma}_{R} \odot \mathrm{THR}^{k}_{N}}.$$

Moreover, restricted to the latter domain they are the same function:

$$(\mathrm{dCollision}^{k,\gamma}_{N,R})^{\leq N} = \mathrm{GapOR}^{\gamma}_{R} \odot \mathrm{THR}^{k}_{N}.$$

We are now ready to give the proof of Lemma 4.4.10.

Proof of Lemma 4.4.10. Using Proposition 4.4.11, instead of $\mathrm{Collision}^{k,\gamma/c}_{N,N}$ we can consider $\mathrm{dCollision}^{k,\gamma}_{N,R}$ (with the appropriate parameters) to show a lower bound. By Observation 4.4.12, we can use Lemma 4.4.7 to relate the query complexity of $\mathrm{dCollision}^{k,\gamma}_{N,R}$ to the double-promise degree of $\mathrm{GapOR}^{\gamma}_{R} \odot \mathrm{THR}^{k}_{N}$ with domain further restricted to $H^{NR}_{\leq N}$. $\square$

4.4.3 Main lower bound

Let us fix $f = (\mathrm{GapOR}^{\gamma}_{R} \odot \mathrm{THR}^{k}_{N})$ with domain $D = D_{(\mathrm{GapOR}^{\gamma}_{R} \odot \mathrm{THR}^{k}_{N})}$ (See Observation 4.4.12). For technical reasons, in the rest of the section, we fix $k \geq 3$ and $N = \lceil 20(2k)^{k/2} \rceil R$.[2]

We first define a construction used to compose dual polynomials, which was introduced in earlier line of work [SZ09, Lee09, She13].

Definition 4.4.13 (Dual block composition). *The dual block composition of two functions $\phi : \{-1,1\}^n \to \mathbb{R}$ and $\psi : \{-1,1\}^m \to \mathbb{R}$ is a function $\phi \star \psi : \{-1,1\}^{nm} \to \mathbb{R}$ defined as*

$$(\phi \star \psi)(x) = 2^n \, \phi(\mathrm{sgn}(\psi(x_1)), \ldots, \mathrm{sgn}(\psi(x_n))) \prod_{i \in [n]} |\psi(x_i)|$$

where $x = (x_1, \ldots, x_n)$ and $x_i \in \{-1,1\}^m$, for $i \in [n]$.

This subsection is dedicated to the proof of the following lemma which, together with Lemma 4.4.10, implies Theorem 4.4.1. Observe that we have to zero out the support of the dual polynomial outside of $H^{NR}_{\leq N}$, since our target domain is not D but $D \cap H^{NR}_{\leq N}$ in Lemma 4.4.10.

Lemma 4.4.14. *Let $N = \lceil 20(2k)^{k/2} \rceil R$ and $0 < \gamma < 1/4^{k-1}$. Then there exists a function $\zeta : \{-1,1\}^{NR} \to \mathbb{R}$ such that*

$$\forall x \in \{-1,1\}^{NR} \setminus H^{NR}_{\leq N}, \quad \zeta(x) = 0; \tag{4.7}$$

$$\sum_{x \in H^{NR}_{\leq N} \cap D} \zeta(x) f(x) - \sum_{x \in H^{NR}_{\leq N} \setminus D} |\zeta(x)| > 2/3; \tag{4.8}$$

$$\|\zeta\|_1 = 1 \quad \textit{and} \quad \mathrm{phd}(\zeta) \in \Omega\left(\sqrt{N^{1-1/k}} / \ln^2 N\right). \tag{4.9}$$

2. These parameters are used in [BKT20] to prove Proposition 4.4.20, which we will use.

Proof. The construction of ζ starts by *block composing* (Definition 4.4.13) two dual polynomials ϕ, ψ, one for GapOR_R^γ and one for THR_N^k. The dual polynomial ϕ for GapOR_R^γ is given by Proposition 4.4.15. The dual polynomial ψ for THR_N^k is given by Proposition 4.4.17.

The block composition $\phi \star \psi$ is a good candidate for the dual polynomial of f. Indeed, Lemma 4.4.23 shows that it satisfies Equation (4.8), showing correlation at least $9/10 > 2/3$. One could also check that it satisfies Equation (4.9). Nonetheless, it does not satisfy Equation (4.7).

We can now use Lemma 4.4.18 to argue that there exists another dual polynomial ζ that satisfies Equation (4.7) and Equation (4.9). Moreover, this ζ is close to $\phi \star \psi$ so that it also satisfies Equation (4.8), with the weaker but sufficient correlation $9/10 - 2/9 > 2/3$. This concludes the proof. $\square$

As we have seen, the previous proof relies on several results, now we are going to zoom on each of them. The first one provides a dual polynomial for OR.

Proposition 4.4.15. *Let $\phi : \{-1,1\}^R \to \mathbb{R}$ be such that $\phi(-1^R) = -1/2$, $\phi(1^R) = 1/2$, and $\phi(z) = 0$ for all $z \in \{-1,1\}^R \setminus \{-1^R, 1^R\}$. Then $\|\phi\|_1 = 1$, $\mathrm{phd}(\phi) \geq 1$, and*

$$\sum_{x \in \{-1,1\}^R} \phi(x)\mathrm{OR}(x) = 1.$$

Proof. $\|\phi\|_1 = |1/2| + |-1/2| = 1$.

For any constant c, $\sum_{x\in\{-1,1\}^R} c \cdot \phi(x) = c/2 - c/2 = 0$, thus $\mathrm{phd}(\phi) \geq 1$.

$\sum_{x\in\{-1,1\}^R} \phi(x)\mathrm{OR}(x) = -1/2 \cdot (-1) + 1/2 \cdot 1 = 1$. $\square$

Next, we look at an explicit dual polynomial for the threshold function and some properties it satisfies.

Definition 4.4.16. *Let $M \in \mathbb{N}$ and $\alpha, \beta > 0$. We say that a function $\omega : [M]_0 \to \mathbb{R}$ satisfies the (α, β)-decay condition if $\sum_{t\in[M]_0} \omega(t) = 0$, $\sum_{t\in[M]_0} |\omega(t)| = 1$ and $|\omega(t)| \leq \alpha e^{-\beta t}/t^2$.*

In [BKT20, Section 5.1] the authors define a dual polynomial ψ of THR_N^k in the following way. Let $k, N \in \mathbb{N}$, and T an integer such that $k \leq T \leq N$. Let $c = 2k\lceil N^{1/k}\rceil$ and $m = \lfloor\sqrt{T/c}\rfloor$. Define set $S = \{1, 2, \ldots, k\} \cup \{ci^2 : 0 \leq i \leq m\}$. Define a univariate polynomial

$$\omega(t) = \frac{(-1)^{t+T-m+1}}{T!}\binom{T}{t}\prod_{r\in[T]_0\setminus S}(t - r).$$

Then let $\psi : \{-1,1\}^N \to \mathbb{R}$ be $\psi(x) = \omega(|x|_H)/\binom{N}{|x|_H}$ for $x \in H_{\leq T}^N$ and $\psi(x) = 0$ otherwise.

Let D_+ and D_- denote the set of false positives and that of false negatives respectively if ψ is considered as a hypothesis for THR_N^k, i.e. $D_+ = \{x \in \{-1,1\}^N : \psi(x) > 0, \mathrm{THR}_N^k(x) = -1\}$ and $D_- = \{x \in \{-1,1\}^N : \psi(x) < 0, \mathrm{THR}_N^k(x) = 1\}$.

They show that ψ and ω have the following properties.

Proposition 4.4.17. *[BKT20, Proposition 5.4] Let ω and ψ be the polynomials defined above. Then the following are true.*

1. $\sum_{x\in D_+} |\psi(x)| \leq \frac{1}{48N}$;

2. $\sum_{x \in D_-} |\psi(x)| \leq \frac{1}{2} - \frac{2}{4^k}$;
3. $\|\psi\|_1 = 1$;
4. $\mathrm{phd}(\psi) \geq c_1\sqrt{k^{-1}TN^{-1/k}}$;
5. *ω satisfies the (α, β)-decay condition with $\alpha = (2k)^k$ and $\beta = c_2/\sqrt{kTN^{1/k}}$.*

The next lemma is already adapted to our functions; it is a consequence of more general results.

Lemma 4.4.18. *Let $N = \lceil 20(2k)^{k/2} \rceil R$, $\phi : \{-1,1\}^R \to \mathbb{R}$ from Proposition 4.4.15 and $\psi : \{-1,1\}^N \to \mathbb{R}$ from Proposition 4.4.17.*

Then there exists a $\zeta : \{-1,1\}^{NR} \to \mathbb{R}$ such that

- $\|\zeta\|_1 = 1$;
- $\mathrm{phd}(\zeta) = \Omega(\sqrt{N^{1-1/k}}/\ln^2 N)$;
- $\|\zeta - \phi \star \psi\|_1 \leq 2/9$;
- *$\zeta(x) = 0$ for all $x \in \{-1,1\}^{NR} \setminus H^{NR}_{\leq N}$.*

Before proving this, we state two propositions from [BKT20] that we are going to use in the proof. The first one is about the properties of the dual block composition.

Proposition 4.4.19. *[BKT20, Proposition 2.20] Let $\phi : \{-1,1\}^n \to \mathbb{R}$, $\psi : \{-1,1\}^m \to \mathbb{R}$. The dual block composition has the following properties.*

1. *If $\|\phi\|_1 = 1$, $\|\psi\|_1 = 1$ and $\langle \psi, 1^{2^m} \rangle = 0$, then $\|\phi \star \psi\|_1 = 1$.*
2. *If $\mathrm{phd}(\phi) \geq \Delta$ and $\mathrm{phd}(\psi) \geq \Delta'$, then $\mathrm{phd}(\phi \star \psi) \geq \Delta \cdot \Delta'$.*

The second one proves the existence of the final dual polynomial ζ, that is close to the "almost good" block composition $\phi \star \psi$, given that some conditions are satisfied.

Proposition 4.4.20. *[BKT20, Proposition 2.22] Let $R \in \mathbb{N}$ sufficiently large and $M \leq R$. Let $\phi : \{-1,1\}^R \to \mathbb{R}$ with $\|\phi\|_1 = 1$, and let $\omega : [M]_0 \to \mathbb{R}$ satisfy the (α, β)-decay condition with some $1 \leq \alpha \leq R^2$ and $4\ln^2 R/(\sqrt{\alpha}R) \leq \beta \leq 1$. Let $N = \lceil 20\sqrt{\alpha} \rceil R$ and $\psi : \{-1,1\}^N \to \mathbb{R}$ be defined as $\psi(x) = \omega(|x|_H)/\binom{N}{|x|_H}$. Let $\Delta < N$ be such that $\mathrm{phd}(\phi \star \psi) \geq \Delta$. Then there exist a $\Delta' \geq \beta\sqrt{\alpha}R/(4\ln^2 R)$ and a function $\zeta : \{-1,1\}^{NR} \to \mathbb{R}$ such that*

1. $\mathrm{phd}(\zeta) \geq \min\{\Delta, \Delta'\}$;
2. $\|\zeta - \phi \star \psi\|_1 \leq 2/9$;
3. $\|\zeta\|_1 = 1$;
4. *$\forall x \in \{-1,1\}^{NR}$ with $|x|_H > N$ $\zeta(x) = 0$.*

Now we can proceed with the proof of the lemma.

Proof of Lemma 4.4.18. From Proposition 4.4.17 (with $T = N$), we know that $\|\psi\|_1 = 1$, and that $\mathrm{phd}(\psi) \geq c_1\sqrt{k^{-1}N^{1-1/k}}$. From Proposition 4.4.15, we know that $\|\phi\|_1 = 1$ and $\mathrm{phd}(\phi) \geq 1$. Using Item 1 of Proposition 4.4.19, we obtain $\|\phi \star \psi\|_1 = 1$, and using Item 2, we get $\mathrm{phd}(\phi \star \psi) \geq c_1\sqrt{k^{-1}N^{1-1/k}}$.

From Proposition 4.4.17, we know that the function ω that is used to define ψ satisfies the (α, β)-decay condition for some constant $\alpha = (2k)^k$ and $\beta = c_2/\sqrt{kN^{1+1/k}}$.

This way, we can use Proposition 4.4.20 to obtain the function ζ we wanted. Indeed, our functions ψ and ϕ satisfy all the conditions of the lemma with pure high degree lower bounded by $\Delta = c_1\sqrt{k^{-1}N^{1-1/k}}$; and with our parameters α and β we obtain $\Delta' = c_2(2k)^{k/2}R/(4\ln^2(R)\sqrt{kN^{1+1/k}}) \in \Omega(\sqrt{N^{1-1/k}}/\ln^2 N)$.

$\square$

For the next lemma, we will use the following proposition, which was implicitly used in the proofs of [BKT20, Propositions 5.5 and 5.6] but not stated in this general form. By convention, we denote $D_{+1} = D_+$ and $D_{-1} = D_-$.

Proposition 4.4.21. *Let $S \subseteq \{-1,1\}^{NR}$. Let $g : \{-1,1\}^R \to \{-1,1\}$, $h : \{-1,1\}^N \to \{-1,1\}$, $\phi : \{-1,1\}^R \to \mathbb{R}$. Let $\psi : \{-1,1\}^N \to \mathbb{R}$ be such that $\|\psi\|_1 = 1$ and $\sum_{x\in\{-1,1\}^N} \psi(x) = 0$. Then the following hold.*

1. *When λ denotes the probability mass function $\lambda(u) = |\psi(u)|$:*

$$\sum_{x\in S} |(\phi \star \psi)(x)| = \sum_{z\in\{-1,1\}^R} |\phi(z)| \cdot \Pr_{x\sim\lambda^{\otimes R}}[x \in S|(\dots, \mathrm{sgn}(\psi(x_i)), \dots) = z].$$

2. *When $\mu_i^{z_i}$ denotes the probability mass function on $\{-1,1\}$ (parameterized by $z_i \in \{-1,1\}$) such that $\mu_i^{z_i}(-1) = 2\sum_{x\in D_{z_i}} |\psi(x)|$, and $\mu = \mu^z = \mu_1^{z_1} \otimes \dots \otimes \mu_R^{z_R}$ the independent product distribution on $\{-1,1\}^R$:*

$$\sum_{x\in\{-1,1\}^{NR}} (\phi \star \psi)(x) \cdot (g \odot h)(x) = \sum_{z\in\{-1,1\}^R} \phi(z) \cdot \mathop{\mathbb{E}}_{y\sim\mu}[g(\dots, y_i z_i, \dots)].$$

Proof. We will need the following claim.

Claim 4.4.22. *Let λ denote the probability mass function $\lambda(u) = |\psi(u)|$ for $u \in \{-1,1\}^N$. Then*

$$\Pr_{u\sim\lambda}[\psi(u) > 0] = \Pr_{u\sim\lambda}[\psi(u) < 0] = \frac{1}{2}.$$

Proof of claim. We know that $\sum_u \psi(u) = 0$. Thus $\sum_{u:\psi(u)>0} |\psi(u)| - \sum_{u:\psi(u)>0} |\psi(u)|$. We then conclude using that $\|\psi\|_1 = 1$. $\diamond$

First part of Proposition 4.4.21 Below, we first apply the definition of the dual block composition (and the fact that 2^R and $\prod_{i\in[R]} |\psi(x_i)|$ are positive). Then we use the definition of λ which ensures that $\prod_{i\in[R]} |\psi(x_i)|$ is the probability of getting $x = (\dots, x_i, \dots)$ when sampling independently R times from distribution λ.

$$\begin{aligned}\sum_{x\in S} |(\phi \star \psi)(x)| &= 2^R \sum_{x\in\{-1,1\}^{NR}} \left(\prod_{i\in[R]} |\psi(x_i)|\right) \cdot |\phi(\dots, \mathrm{sgn}(\psi(x_i)), \dots)| \cdot \mathbb{I}[x \in S] \\ &= 2^R \cdot \mathbb{E}_{x\sim\lambda^{\otimes R}}[|\phi(\dots, \mathrm{sgn}(\psi(x_i)), \dots)| \cdot \mathbb{I}[x \in S]]\end{aligned}$$

We introduce new variables z_i that will be compared to $\mathrm{sgn}(\psi(x_i))$. Using Claim 4.4.22, the probability of picking a $z \in \{-1,1\}^R$ from the uniform distribution such that z corresponds to the vector of the signs is $\frac{1}{2^R}$. Thus, the previous term can be rewritten as

$$\begin{aligned} & 2^R \sum_{z\in\{-1,1\}^R} |\phi(z)| \cdot \Pr_{x\sim\lambda^{\otimes R}}[x \in S \wedge (\dots, \mathrm{sgn}(\psi(x_i)), \dots) = z] \\ = & \sum_{z\in\{-1,1\}^R} |\phi(z)| \cdot \Pr_{x\sim\lambda^{\otimes R}}[x \in S \mid (\dots, \mathrm{sgn}(\psi(x_i)), \dots) = z] \end{aligned}$$

which completes the proof.

Second part of Proposition 4.4.21 Remember that λ denotes the probability mass function $\lambda(u) = |\psi(u)|$ for $u \in \{-1,1\}^N$. Just like in the proof of the first item,

$$\begin{aligned} & \sum_{x\in\{-1,1\}^{NR}} (\phi \star \psi)(x) \cdot (g \odot h)(x) \\ = & \sum_{z\in\{-1,1\}^R} \phi(z) \cdot \mathop{\mathbb{E}}_{x\sim\lambda^{\otimes R}} [(g \odot h)(x) \mid (\dots, \mathrm{sgn}(\psi(x_i)), \dots) = z]\,. \end{aligned}$$

Using Claim 4.4.22, we can first notice that for any $b \in \{-1,1\}$, the probability that an x_i sampled from λ is a false b (i.e. false positive if $b = 1$ and false negative if $b = -1$) is as follows, where, by convention, $D_{+1} = D_+$ and $D_{-1} = D_-$:

$$\begin{aligned} \Pr_{x_i\sim\lambda}\Big[h(x_i) \neq \mathrm{sgn}(\psi(x_i)) \mid \mathrm{sgn}(\psi(x_i)) = b\Big] &= \sum_{x_i\in D_b} \Pr_{x_i\sim\lambda}\Big[\text{sampling } x_i \mid \mathrm{sgn}(\psi(x_i)) = b\Big] \\ &= 2\sum_{x_i\in D_b} |\psi(x_i)|. \end{aligned}$$

Therefore, if $z_i = \mathrm{sgn}(\psi(x_i))$ and x_i is a false z_i, it means that z_i should be flipped to get $h(x_i)$. Let $y_i \in \{-1,1\}$ denote whether we flip z_i. As x_i is a false z_i with probability $2\sum_{x_i\in D_{z_i}} |\psi(x_i)|$, this is the probability with which we should flip z_i, i.e. the probability that $y_i = -1$.

Thus, for any $z \in \{-1,1\}^R$, the vector $(\dots, h(x_i), \dots)$ with $x \sim \lambda^{\otimes R}$ conditioned on $(\dots, \mathrm{sgn}(\psi(x_i)), \dots) = z$ is identically distributed with $(\dots, z_i y_i, \dots)$ where y_i are random bitflips according to $\mu_i^{z_i}$: $y_i = -1$ with probability $2\sum_{x_i\in D_{z_i}} |\psi(x_i)|$ and $y_i = 1$ otherwise.

Now we can finish the proof:

$$\begin{aligned} & \sum_{z\in\{-1,1\}^R} \phi(z) \cdot \mathop{\mathbb{E}}_{x\sim\lambda^{\otimes R}} [(g \odot h)(x) \mid (\dots, \mathrm{sgn}(\psi(x_i)), \dots) = z] \\ = & \sum_{z\in\{-1,1\}^R} \phi(z) \cdot \mathop{\mathbb{E}}_{y\sim\mu} [g(\dots, z_i y_i, \dots)]. \end{aligned}$$

□

Finally, we are ready to prove the last missing statement, which is our main technical contribution to this part. The proof of [BKT20, Lemma 6.9] does not apply directly to this problem: they use the fact that the dual polynomial ψ of their inner function (OR) has one sided error, which is not the case here.

As now we focus on the composed function f (and the dual composition $\phi \star \psi$), the domain is not restricted to small Hamming weight inputs anymore.

Lemma 4.4.23. *Let $N = \lceil 20(2k)^{k/2} \rceil R$ and $0 < \gamma < 1/4^{k-1}$. Functions ϕ from Proposition 4.4.15 and ψ from Proposition 4.4.17 satisfy*

$$\sum_{x \in D} (\phi \star \psi)(x) \cdot f(x) - \sum_{x \in \{-1,1\}^{NR} \setminus D} |(\phi \star \psi)(x)| \geq 9/10.$$

Proof. We rewrite the left-hand side by manipulating the sets we consider in the sums, and then we will bound separately the terms we get.

$$\begin{aligned}
&\sum_{x \in D} (\phi \star \psi)(x) \cdot f(x) - \sum_{x \in \{-1,1\}^{NR} \setminus D} |(\phi \star \psi)(x)| \\
&= \sum_{x \in \{-1,1\}^{NR}} (\phi \star \psi)(x) \cdot (\mathrm{OR} \odot \mathrm{THR}^k_N)(x) \\
&\quad - \left(\sum_{x \in \{-1,1\}^{NR} \setminus D} (\phi \star \psi)(x) \cdot (\mathrm{OR} \odot \mathrm{THR}^k_N)(x) + \sum_{x \in \{-1,1\}^{NR} \setminus D} |(\phi \star \psi)(x)| \right) \\
&\geq \sum_{x \in \{-1,1\}^{NR}} (\phi \star \psi)(x) \cdot (\mathrm{OR} \odot \mathrm{THR}^k_N)(x) - 2 \sum_{x \in \{-1,1\}^{NR} \setminus D} |(\phi \star \psi)(x)|
\end{aligned}$$

We first lower bound the first term.

Claim 4.4.24.

$$\sum_{x \in \{-1,1\}^{NR}} (\phi \star \psi)(x) \cdot (\mathrm{OR} \odot \mathrm{THR}^k_N)(x) \geq 1 - e^{-\frac{R}{4^{k-1}}} - \frac{R}{48N}.$$

Proof of claim. Using Item 2 of Proposition 4.4.21, the left-hand side can be written as

$$\sum_{z \in \{-1,1\}^R} \phi(z) \cdot \mathop{\mathbb{E}}_{y \sim \mu} [\mathrm{OR}(\ldots, y_i z_i, \ldots)].$$

Recall that $\phi(z) = 0$ when z is anything but -1^R or 1^R, so only two terms are left to study.

If $z = -1^R$, using Item 2 of Proposition 4.4.17, each y_i is -1 with probability $\leq 1 - 1/4^{k-1}$ and 1 with probability $\geq 1/4^{k-1}$. If there is any $y_i = 1$, then the value of the OR is still -1. The probability of this event is $\geq 1 - (1 - 1/4^{k-1})^R \geq 1 - e^{-\frac{R}{4^{k-1}}}$. So, the expected value is $\leq (-1)(1 - e^{-\frac{R}{4^{k-1}}}) + e^{-\frac{R}{4^{k-1}}} = -1 + 2e^{-\frac{R}{4^{k-1}}}$. Since in this case $\phi(-1^R) = -1/2$, the contribution to the sum is at most $1/2 - e^{-\frac{R}{4^{k-1}}}$.

If $z = 1^R$, then, using Item 1 of Proposition 4.4.17, each y_i is -1 with probability $\leq 1/(48N)$. If any y_i is -1, then the value of the OR becomes -1. The union bound tells us that the probability of this is $\leq R/(48N)$, so the expected value is at least $-R/(48N) + 1 - R/(48N) = 1 - R/(24N)$. Multiplied by $\phi(1^R) = 1/2$, the contribution is at least $1/2 - R/(48N)$. Thus, the first term can be lower bounded by $1 - e^{-\frac{R}{4^{k-1}}} - \frac{R}{48N}$. ◇

Now we bound the second term.

Claim 4.4.25.
$$2 \sum_{x \in \{-1,1\}^{NR} \setminus D} |(\phi \star \psi)(x)| < e^{-2R\left(\frac{1}{4^{k-1}} - \gamma\right)^2}.$$

Proof of claim. By Item 1 of Proposition 4.4.21 with $S = \{-1,1\}^{NR} \setminus D$, the term can be written as follows,

$$2 \sum_{x \in \{-1,1\}^{NR} \setminus D} |(\phi \star \psi)(x)| = 2 \sum_{z \in \{-1,1\}^{R}} |\phi(z)| \cdot \Pr_{x \sim \lambda^{\otimes R}} [x \notin D \mid (\dots, \operatorname{sgn}(\psi(x_i)), \dots) = z],$$

which, using that $|\phi(z)| = 1/2$ when z is -1^R or 1^R and 0 otherwise, collapses to

$$\Pr_{x \sim \lambda^{\otimes R}} [x \notin D \mid (\dots, \operatorname{sgn}(\psi(x_i)), \dots) = -1^R] + \Pr_{x \sim \lambda^{\otimes R}} [x \notin D \mid (\dots, \operatorname{sgn}(\psi(x_i)), \dots) = 1^R].$$

In order to bound these two terms, we introduce $0/1$-variables r_i and q_i, for $i \in [R]$, related to the false positive and false negative inputs. Define $r_i = 1$ if $\mathrm{THR}^k_N(x_i) = -1$ and $\operatorname{sgn}(\psi(x_i)) = 1$, and otherwise $r_i = 0$. Similarly, $q_i = 1$ if $\mathrm{THR}^k_N(x_i) = 1$ and $\operatorname{sgn}(\psi(x_i)) = -1$, and otherwise $q_i = 0$.

Let us focus on the first term. If we sample x_i from the conditional distribution $(\lambda | \operatorname{sgn}(\psi(x_i)) = 1)$, then

$$\Pr[r_i = 1] = \Pr\left[\mathrm{THR}^k_N(x_i) = -1 | \operatorname{sgn}(\psi(x_i)) = 1\right] = 2 \sum_{x_i \in D_+} |\psi(x_i)| \leq 1/(24N),$$

where in the last step we used Item 1 of Proposition 4.4.17. Thus, we can upper bound the probability that an input does not satisfy the promise of $\mathrm{GapOR}^{\gamma}_R$ (i.e. that it is not in D) knowing that all the predictions are 1. It means that it contains at least one but less than γR many -1s, so this many predictions are false positive, which can be expressed by the r_i variables. In the last step below, we use the union bound.

$$\begin{aligned}\Pr\left[x \notin D \mid \forall i \in [R] \operatorname{sgn}(\psi(x_i)) = 1\right] &= \Pr\left[1 \leq \sum_{i \in [R]} r_i < \gamma R\right] \\ &\leq \Pr\left[1 \leq \sum_{i \in [R]} r_i\right] \leq \frac{R}{24N}.\end{aligned}$$

Similarly, for the second term, if we sample x_i from the conditional distribution $(\lambda | \operatorname{sgn}(\psi(x_i)) = -1)$, then

$$\Pr[q_i = 1] = \Pr\left[\mathrm{THR}^k_N(x_i) = 1 | \operatorname{sgn}(\psi(x_i)) = -1\right] = 2 \sum_{x_i \in D_-} |\psi(x_i)| \leq 1 - \frac{1}{4^{k-1}},$$

where in the last step we used Item 2 of Proposition 4.4.17.

Then, similarly to the first term, we can upper bound the probability. Now in the last step we use Hoeffding's inequality (Corollary 2.4.2), which introduces the constraint $\gamma < \frac{1}{4^{k-1}}$.

$$\Pr\left[x \notin D \mid \forall i \in [R] \operatorname{sgn}(\psi(x_i)) = -1\right] \leq \Pr\left[(1-\gamma)R < \sum_{i \in [R]} q_i\right] < e^{-2R\left(\frac{1}{4^{k-1}} - \gamma\right)^2}.$$

◇

Putting together the two bounds, we obtain

$$\sum_{x\in D}(\phi\star\psi)(x)\cdot f(x)-\sum_{x\in\{-1,1\}^{NR}\setminus D}|(\phi\star\psi)(x)|\geq 1-\frac{R}{16N}-e^{-\frac{R}{4^{k-1}}}-e^{-2R(\frac{1}{4^{k-1}}-\gamma)^2}.$$

When k and $1/4^{k-1}-\gamma$ are positive constants and $R\in\Theta(N)$, this is larger than $9/10$ (for large enough N). □

Finally, we can conclude the proof of Theorem 4.4.1.

Proof of Theorem 4.4.1. By Lemma 4.4.14, there is a dual polynomial for $\mathrm{GapOR}_R^{\gamma}\odot\mathrm{THR}_N^k$ of pure high degree $\Omega(\sqrt{N^{1-1/k}}/\ln^2 N)$, that is only supported on $H_{\leq N}^{NR}$. By Theorem 4.4.8, this means that the double-promise δ-approximate degree of $\mathrm{GapOR}_R^{\gamma}\odot\mathrm{THR}_N^k$, with domain restricted to $H_{\leq N}^{NR}$, is $\Omega(\sqrt{N^{1-1/k}}/\ln^2 N)$. Using Lemma 4.4.10 with $c=\lceil 20(2k)^{k/2}\rceil$, we obtain that the bounded-error quantum query complexity of $\mathrm{Collision}_{N,N}^{k,\gamma'}$ is $\Omega(\sqrt{N^{1-1/k}}/\ln^2 N)$ if $\gamma'=\gamma/c<1/(4^{k-1}\lceil 20(2k)^{k/2}\rceil)$. This implies the same lower bound on testing k-collision-freeness with $\varepsilon=\gamma'$, as Collision is just a more restricted version of the same problem. □

4.5 Testing 3-colourability

Let $G=(V,E)$ be an undirected graph on n vertices. For positive integer $k\leq n$, a *k-colouring* of G is a function $c:V\to[k]$ (so it is a vertex colouring). A k-colouring is called *proper* if $\forall\{u,v\}\in E:\ c(u)\neq c(v)$. In words, a k-colouring assigns one of k available colours to each vertex of G, and in a proper colouring the two endpoints of each edge in G have different colours. We call a graph *k-colourable* if it has a proper k-colouring.

In this section, we will prove that the problem of property testing 3-colourability in bounded degree graphs remains maximally hard-to-test in the quantum setting. Our lower bound proof will roughly follow the same approach as that of [BOT02]. See [BY22, Section 5.6] also for a reference.

Theorem 4.5.1 (Restatement of Theorem 4.1.4)**.** *Let G be an unknown undirected N-vertex graph with maximum degree d, and $\varepsilon\in(0,1)$ be a parameter. Given quantum query access to G in the undirected bounded-degree graph model, in order to distinguish if G is 3-colourable, or if it is ε-far from being 3-colourable, $\Omega(N)$ quantum queries are necessary.*

In order to prove the above theorem, we will first discuss the approach to proving the classical lower bound. Then we will modify the classical proof suitably to the quantum setting. Let us start with the notion of k-wise independent string which will be used both in the classical and quantum lower bound proofs.

Definition 4.5.2 (k-wise independent string)**.** *Let $S\subseteq\{0,1\}^N$ and let string $s=(s_1,\ldots,s_N)\in\{0,1\}^N$ be chosen uniformly at random from S. The string s is said to be* k-wise independent *if for any set of k-indices $i_1,i_2\ldots,i_k$, the probability of any particular assignment $(b_{i_1},b_{i_2},\ldots,b_{i_k})\in\{0,1\}^k$ to the indices $i_1,i_2\ldots,i_k$ is equal to $1/2^k$.*

4.5.1 Classical lower bound approach for testing 3-colourability

As we discussed in the technical overview, to prove the lower bound of 3-colourability, the authors in [BOT02] studied another problem called E$(3, c)$LIN-2, a problem related to deciding the satisfiability of a system of linear equations. Then the authors designed a reduction to 3-colourability from E$(3, c)$LIN-2, which finally proves the linear query complexity lower bound for testing 3-colourability. We will also follow a similar approach here. Let us first formally define the problem of E$(3, c)$LIN-2.

Definition 4.5.3 (E$(3, c)$LIN-2)**.** *Let $\mathcal{E}$ be a system of linear equations with N variables from $\mathbb{F}_2$, where there are 3 variables in each equation, and each variable occurs in at most c equations. This system $\mathcal{E}$ is represented as a matrix-vector pair and we have query access to their entries. Given a parameter $\alpha \in (0, 1)$, the goal is to distinguish if $\mathcal{E}$ is satisfiable, or at least an α-fraction of the equations need to be modified to make $\mathcal{E}$ satisfiable.*

The authors in [BOT02] proved the following lemma, which states that there exists a system of linear equations (equivalently a matrix), such that any constant fraction of the rows of this matrix are linearly independent. The authors proved this using hypergraph constructions.

Lemma 4.5.4 ([BOT02, Theorem 8])**.** *For every $c > 0$, there exists a $\delta > 0$ such that for every N, there exists a matrix $A \in \{0, 1\}^{cN\times N}$ with cN rows and N columns such that the following conditions hold:*

1. *Each row of A has exactly three non-zero entries.*
2. *Each column of A has exactly $3c$ non-zero entries.*
3. *Every collection of $\delta \cdot N$ rows of A is linearly independent.*

Using the existence of the matrix A corresponding to Lemma 4.5.4, the authors in [BOT02] used Yao's minimax principle [Yao77] to prove a linear lower bound for testing E$(3, c)$LIN-2. Using this technique allows one – in some cases – to prove a lower bound on the worst-case complexity of a probabilistic algorithm, by instead considering the best performance of a deterministic algorithm over the hardest distribution of the inputs. For this problem, they designed a pair of hard-to-distinguish distributions D_{yes} and D_{no}, such that, unless $\Omega(N)$ queries are performed, no algorithm can distinguish between them. We present this construction in the proof sketch of the following lemma.

Lemma 4.5.5. *There exists a matrix $A \in \{0, 1\}^{cN\times N}$ (similar to the matrix mentioned in Lemma 4.5.4) such that given a parameter $\varepsilon \in (0, 1)$ and query access to A and a vector $y \in \{0, 1\}^{cN}$, in order to distinguish if there exists another vector $x \in \{0, 1\}^N$ such that $Ax = y$, or for any vector $x \in \{0, 1\}^N$ only a constant ε fraction of the constraints encoded by A and y are satisfied, $\Omega(N)$ queries are necessary.*

Proof sketch. As we mentioned, this proof follows Yao's minimax lower bound technique. A pair of hard distributions D_{yes} and D_{no} are constructed, such that, unless $\Omega(N)$ queries are performed, no algorithm can distinguish between them.

Let us consider the matrix $A \in \{0, 1\}^{cN\times N}$ as mentioned in Lemma 4.5.4. Based on the matrix A, the hard-to-distinguish distributions D_{yes} and D_{no} are as follows:

1. D_{yes}: Choose a vector $z \in \{0, 1\}^N$ uniformly at random from $\{0, 1\}^N$, and set the vector $y \in \{0, 1\}^{cN}$ as $y = Az$. Then the system of linear equations is $Ax = y$.

2. D_{no}: Choose the vector $y \in \{0,1\}^{cN}$ uniformly at random from $\{0,1\}^{cN}$, and set the system of linear equations $Ax = y$.

Now we have the following claim describing the properties of D_{yes} and D_{no}.

Claim 4.5.6.

(i) The system of linear equations corresponding to D_{yes} is satisfiable.

(ii) With probability at least $2/3$, the system of linear equations corresponding to D_{no} is $(1/2-\alpha)$-far from being satisfiable for every $\alpha > 0$.

Note that the system of linear equations in D_{yes} is satisfiable by setting $x = z$. On the other hand, for the system of linear equations corresponding to D_{no}, vector y is uniformly random. Thus, with high probability, vector $Az - y$ has large Hamming weight for any $z \in \{0,1\}^N$, and therefore the system of linear equations $Ax = y$ is far from being satisfiable. The formal proof is in [BOT02, Lemma 18]. □

The authors in [BOT02] proved the following lower bound for testing E$(3,c)$LIN-2.

Lemma 4.5.7 ([BOT02, Lemma 19])**.** *For every $\alpha > 0$, there are constants c and $\delta > 0$ such that every algorithm that distinguishes satisfiable instances of E$(3,c)$LIN-2 with N variables from instances that are $(1/2-\alpha)$-far from satisfiable must have classical query complexity at least δN.*

The key insight that is used to prove the above lemma is the following. In the case of D_{no}, vector y is uniformly random. On the other hand, in the case of D_{yes}, applying Lemma 4.5.4, any δN rows of A are linearly independent, thus any subset of δN entries of $y = Az$ will look uniformly random. Hence, y is k-wise independent with $k = \delta N$. It remains to use the fact that it takes $\Omega(k)$ queries to distinguish a k-wise independent vector from a uniformly random one. We will not formally prove the above lemma here, please refer to [BOT02] for a formal proof.

Finally, we have the reduction that maps satisfying instances of testing E$(3,c)$LIN-2 to satisfying instances of testing 3-colourability and vice-versa.

Lemma 4.5.8 ([BOT02, Section 4])**.** *There exists a reduction φ that maps instances of testing E$(3,c)$LIN-2 to instances of testing 3-colourability such that the following hold:*

1. *If an input x to E$(3,c)$LIN-2 is satisfiable, then $\varphi(x)$ is a 3-colourable graph.*
2. *If an input x to E$(3,c)$LIN-2 is far from being satisfiable, then $\varphi(x)$ is a graph that is far from being 3-colourable.*

4.5.2 Quantum lower bound for testing E$(3,c)$LIN-2 and 3-colourability

We will first prove the quantum lower bound for testing E$(3,c)$LIN-2. Our result is stated as follows.

Lemma 4.5.9. *For every $\alpha > 0$, there are constants c and $\delta > 0$, such that every algorithm that distinguishes satisfiable instances of E$(3,c)$LIN-2 with N variables from instances that are $(1/2-\alpha)$-far from satisfiable must have quantum query complexity at least $\delta N/2$.*

In order to prove the above theorem, we will be using the following well-known result, which states that distinguishing between a uniformly random string and an ℓ-wise independent string, for an appropriate integer ℓ, is hard for quantum algorithms.

Proposition 4.5.10 (see e.g. [ADW22, Fact 1])**.** *The output distribution of a quantum algorithm making q queries to a uniformly random string is identical to the same algorithm making q queries to a $2q$-wise independent string.*

Now let us prove Lemma 4.5.9.

Proof of Lemma 4.5.9. Following Lemmas 4.5.4 and 4.5.5, we know that there exists a matrix A whose δN rows are linearly independent, for which testing $\mathrm{E}(3, c)$LIN-2 requires $\Omega(N)$ classical queries. Moreover, from Proposition 4.5.10, we know that any quantum algorithm that performs less than $k/2$ queries, cannot distinguish a uniformly random vector from a k-wise independent vector. Now let us set $k = \delta N$. Combining all the above, this implies that at least $\delta N/2$ quantum queries are necessary for testing $\mathrm{E}(3, c)$LIN-2. □

Now we are finally ready to prove Theorem 4.5.1.

Proof of Theorem 4.5.1. From Lemma 4.5.9, we know that the quantum query complexity of testing $\mathrm{E}(3, c)$LIN-2 is $\Omega(N)$. In order to prove similar lower bound for testing 3-colourability, we will again use a reduction approach. Given a pair of hard instances corresponding to testing $\mathrm{E}(3, c)$LIN-2, we will apply the reduction φ mentioned in Lemma 4.5.8. Similarly to the classical setting, φ will map the yes instances of $\mathrm{E}(3, c)$LIN-2 to instances of 3-colourable graphs and vice-versa. So, the quantum query lower bound of $\Omega(N)$ carries forward from $\mathrm{E}(3, c)$LIN-2 to 3-colourability. Thus, we conclude that $\Omega(N)$ quantum queries are necessary to test 3-colourability in the bounded degree model. □

4.5.3 Other maximally hard-to-test problems

As we mentioned in the introduction, there are several other problems in the bounded degree graph model, which are maximally hard to test classically. Moreover, their lower bounds stem from similar ideas as the $\mathrm{E}(3, c)$LIN-2 and 3-colourability lower bounds, as mentioned in [YI10a, Gol25]. Following the same path as in the previous subsection, we also obtain $\Omega(N)$ quantum query lower bounds for all these problems. For brevity, we only present the theorem statements below and omit their proofs.

Theorem 4.5.11 (Hamiltonian Path/Cycle)**.** *Given quantum query access to an unknown undirected (directed) d-bounded degree N-vertex graph G for some integer d, and a parameter $\varepsilon \in (0, 1)$, in order to distinguish if G has an undirected (directed) Hamiltonian path/cycle or is ε-far from having an undirected (directed) Hamiltonian path/cycle, $\Omega(N)$ quantum queries are necessary.*

Theorem 4.5.12 (Approximating Independent Set/Vertex Cover size)**.** *Given query access to an unknown undirected d-bounded degree N-vertex graph G for some integer d, and a parameter $\varepsilon \in (0, 1)$, for approximating the independent set size/vertex cover of G, $\Omega(N)$ quantum queries are necessary.*

4.6 A maximally hard-to-test property in the dense graph model

In this section, we are going to give a property that is maximally hard to test in the dense graph model, even for quantum algorithms. Before proceeding to presenting our result, let us first have a brief reminder of the model. Here a graph $G = (V, E)$ with $|V| = N$ is represented as an adjacency matrix A_G. The query access to A_G is defined as follows: For a pair of vertices $u, v \in V$,

$$A_G(u, v) = \begin{cases} 1, & \text{there is an edge between vertices } u \text{ and } v; \\ 0, & \text{otherwise.} \end{cases}$$

A graph G is said to be ε*-far* from some property $\mathcal{P}$ for some parameter $\varepsilon \in (0, 1)$, if one needs to modify (add or remove) at least εN^2 edges of G. Modifying edges is equivalent to changing entries of the adjacency matrix A_G associated to G. Since in the dense graph model G is represented by its adjacency matrix of size $\Theta(N^2)$, any property is testable by performing $O(N^2)$ queries.

Now we proceed to proving that there exists a property that is maximally hard to test quantumly in the adjacency matrix model. In particular, given query access to the adjacency matrix of an unknown undirected graph G, there exists a property that requires $\Omega(N^2)$ quantum queries to test. This is an adaptation of the classical $\Omega(N^2)$ lower bound from [GKNR12, Appendix A]: we show that testing the same property is also hard in the quantum case. Formally, our result is stated as follows.

Theorem 4.6.1. *Let G be an unknown undirected dense graph on N vertices, and $\varepsilon \in (0, 1)$ be a parameter. Given quantum query access to the adjacency matrix of G, there exists a property $\mathcal{P}$ such that $\Omega(N^2)$ quantum queries are necessary to distinguish if G satisfies $\mathcal{P}$, or it is ε-far from satisfying $\mathcal{P}$.*

Let us start by describing the property considered in [GKNR12].

Property $\mathcal{P}$:

Let N and n be integers such that $n = \binom{N}{2}$. This way, we can fix any bijection between sets $[n]$ and $\binom{[N]}{2}$ so that expressions $\{i, j\}$ (for $i, j \in [N]$, $i \neq j$) and $\ell \in [n]$ are interchangeable. Consider a subset $\mathcal{C} \subset \{0, 1\}^n$ of size $|\mathcal{C}| = 2^{n/100}$, such that there exists some parameter $\delta \in (0, 1)$, that a uniformly random $X \in \mathcal{C}$ is (δn)-wise independent. This kind of construction exists in the literature (see e.g. [ABI86, Proof of Proposition 6.5.]) and is based on BCH codes. Without going into details about codes, we note that membership in set $\mathcal{C}$ corresponds to being a codeword of a code, and it is efficiently checkable using the parity-check matrix.

The hard-to-test graphs are constructed in three phases based on this set $\mathcal{C}$:

(i) Let us consider a Boolean string $X \in \{0, 1\}^n$, and let us define an associated graph $G_1(X) = ([N], E_1(X))$, such that for any two indices $i, j \in [N]$, $i \neq j$ there is an edge between vertices i and j of $G_1(X)$ if and only if the $\{i, j\}$-th bit of X is 1. The graph $G_1(X)$ is said to be *good* if the corresponding string $X \in \mathcal{C}$, and it is said to be *bad* if $X \notin \mathcal{C}$.

(ii) Now for any a Boolean string $X \in \{0,1\}^n$, we construct another graph $G_2(X)$ based on $G_1(X)$. We first take a disjoint union of $G_1(X)$ and a clique C on $(2N+1)$ vertices, and then we add some edges between $G_1(X)$ and C. In particular, for all $i \in [N]$, let us add an edge between the i-th vertex of $G_1(X)$ and each of the first i vertices of C. We denote $G_2(X) = ([3N+1], E_2(X))$.

(iii) Finally, since we want to obtain a graph property, we have to ensure invariance over any permutation of the vertices. To any Boolean string $X \in \{0,1\}^n$, let us associate a collection of final graphs $G_{\mathsf{final}}(X)$ by taking the permutation closure of $G_2(X)$. Let S_{3N+1} denote the permutation group on $3N+1$ elements; then $G_{\mathsf{final}}(X)$ is the following set.

$$G_{\mathsf{final}}(X) = \{G_3(X,\sigma) : \sigma \in S_{3N+1}\}$$

Where $G_3(X,\sigma) = ([3N+1], E_3(\sigma))$ is the graph we get from $G_2(X)$ after applying permutation σ on its vertices, i.e. an edge $\{i,j\} \in E_3(\sigma)$ if and only if $\{\sigma^{-1}(i), \sigma^{-1}(j)\} \in E_2(X)$.

Now let us describe the hard-to-distinguish graphs.

1. **Final good graph:** A graph H on $3N+1$ vertices is said to be *final good* if there is a string $X \in \mathcal{C}$ such that H is in the final graph set of X, i.e. $H \in G_{\mathsf{final}}(X)$.
2. **Final bad graph:** A graph H is called a *final bad* graph if there is a string $X \in \{0,1\}^n$ such that $H \in G_{\mathsf{final}}(X)$, but H is not a final good graph.

Property $\mathcal{P}$ is defined as the set of final good graphs.

Classical lower bound approach for testing $\mathcal{P}$:

The authors in [GKNR12] used Yao's minimax lemma [Yao77] to prove the lower bound. Namely, they designed two distributions D_{yes} and D_{no} over graphs on $3N+1$ vertices. The distribution D_{yes} is defined by taking $X \in \mathcal{C}$ uniformly at random and performing phases (i) and (ii) on it. On the other hand, D_{no} is defined by taking a uniformly random $X \in \{0,1\}^n$ and then performing phases (i) and (ii) on this bitsring. Notice that D_{yes} is only supported on final good graphs while D_{no} is supported on both final good and final bad graphs.

Then, they prove the following two lemmas:

Lemma 4.6.2 ([GKNR12, Claim 7.1]). *Any graph G drawn from D_{no} is 0.01-far from $\mathcal{P}$ with probability higher than $9/10$.*

Lemma 4.6.3 ([GKNR12, Claim 7.2]). *Any randomised algorithm that performs $o(N^2)$ queries cannot distinguish graphs drawn from either D_{yes} or D_{no}.*

The proof of the above lemma relies on two facts. (1) If an algorithm could tell apart graphs drawn from D_{yes} or D_{no}, then it could also distinguish $X \in \mathcal{C}$ from $X \in \{0,1\}^n$. (2) Any randomised algorithm that performs at most q queries cannot distinguish between a q-wise independent string, and a uniformly random one.

Quantum lower bound for testing $\mathcal{P}$:

Proposition 4.5.10 tells us that any quantum algorithm that makes at most q queries, cannot distinguish between a uniformly random string, and a $2q$-wise independent string.

Thus, the approach of [GKNR12] works in our context as well, the only difference in the lower bound is a factor of 2. Therefore, we have the following lemma. The proof of this lemma is direct and omitted.

Lemma 4.6.4. *Any quantum algorithm that performs $o(N^2)$ queries cannot distinguish graphs drawn from either D_{yes} or D_{no}.*

Proof of Theorem 4.6.1. Let us consider distributions D_{yes} and D_{no} described above. By construction, any graph drawn from D_{yes} satisfies $\mathcal{P}$, and by Lemma 4.6.2, we know that a graph drawn from D_{no} is far from $\mathcal{P}$ with very high probability. Moreover, from Lemma 4.6.4, we can say that any graphs drawn from either D_{yes} and D_{no} are hard to distinguish. This completes the proof of Theorem 4.6.1. $\square$

Chapter 5

Conclusion and perspectives

This thesis investigates different quantum and classical algorithmic techniques in the realms of approximation algorithms and property testing, with a strong emphasis on applications to topological data analysis and subgraph detection problems. Our contributions span the design of new classical and quantum algorithms, and the establishment of quantum query complexity lower bounds. At the same time, the results presented in this thesis open up several avenues for further research.

The first contribution presented in the thesis focuses on of Betti number approximation in simplicial complexes, a central task in topological data analysis. Prior work had established a quantum algorithm with polynomial dependence on all key parameters, including the number of vertices, the inverse precision and the inverse spectral gap of the normalised combinatorial Laplacian. We provide the first efficient classical alternative under more restrictive assumptions. Our algorithm, based on a path integral Monte Carlo method, relies on approximating traces of matrix powers related to the combinatorial Laplacian. Though limited to regimes where the precision and spectral gap are constant, it serves as a useful classical benchmark against which quantum speedups can be rigorously evaluated. We also refine the analysis for clique complexes, where sparsity can be exploited to broaden the tractable regime.

Our classical algorithm still falls short of matching quantum performance in some parameter regimes, even in the special case of clique complexes. It would be valuable to investigate whether there exist efficient classical algorithms in this regime, or there is an exponential quantum speedup for a certain set of parameters. Another direction of future research could be to rigorously check if our algorithm can be used for estimating persistent Betti numbers with similar efficiency.

In the second contribution, we explore the property testing version of Betti number estimation for clique complexes in the dense graph model. Here, we provide an algorithm that distinguishes whether the k-th Betti number is near-maximal, or the complex is far from this, with query complexity independent of the input size. This is done via a reduction to tolerant property testing of clique-freeness, using a matroidal characterisation of independence, and the well-known graph removal lemma.

This result could potentially be generalised in several ways. Currently, parameter k is required to be constant, and only the property of having an extremely large Betti number over $\mathbb{F}_2$ is shown to be testable. Maybe one could allow larger k, and test having smaller Betti numbers over different rings. While our results are tailored to clique complexes, real-world data often give rise to general simplicial complexes. It would be intriguing to explore whether similar results are possible in this case. For this, an appropriate framework of property testing simplicial complexes would be necessary.

The last main chapter of the thesis addresses quantum property testing of subgraph-freeness in bounded-degree directed graphs, with a focus on the k-collision and k-star problems. We present a quantum algorithm for property testing subgraph-freeness for a large family of directed subgraphs, generalising a prior k-collision finding technique. Complementing this, we prove a quantum lower bound for property testing k-collision-freeness using the dual polynomial method. Our work refines the polynomial construction and correlation analysis, extending previous techniques to new settings. This potentially broadens the class of problems where such lower bounds can be proved. Finally, we show that both in the bounded-degree model and in the dense model, there exist graph problems that are maximally hard to test.

The most obvious gap that our results leave is that the upper and lower bounds do not match, and it would be exciting to see what the true complexity of these problems is. It would also be interesting to see if it is possible to use less quantum memory in our algorithm, in exchange for a slightly increased query complexity. The dual polynomial method remains one of the few tools available for proving quantum lower bounds. However, it is technically demanding and problem-specific. One key challenge is to develop more modular or automatable constructions. A general quantum version of the proportional moments technique [RRSS09] could be one such option. Finally, because of the close connection of the two models, we hope that some new results in the query complexity model could solve open questions in distribution testing.

Bibliography

[AAI$^+$16] Scott Aaronson, Andris Ambainis, Janis Iraids, Martins Kokainis, and Juris Smotrovs. Polynomials, quantum query complexity, and Grothendieck's inequality. In *Proceedings of the 31st Conference on Computational Complexity (CCC)*, volume 50, pages 25:1–25:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. DOI: 10.4230/LIPIcs.CCC.2016.25.

[Aar02] Scott Aaronson. Quantum lower bound for the collision problem. In *Proceedings of 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 635–642. Association for Computing Machinery, 2002. DOI: 10.1145/509907.509999.

[ABC$^+$24] Ismail Yunus Akhalwaya, Ahmed Bhayat, Adam Connolly, Steven Herbert, Lior Horesh, Julien Sorci, and Shashanka Ubaru. Comparing quantum and classical Monte Carlo algorithms for estimating betti numbers of clique complexes. *arXiv:2408.16934*, 2024. DOI: 10.48550/arXiv.2408.16934.

[ABI86] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. DOI: 10.1016/0196-6774(86)90019-2.

[ABRW16] Andris Ambainis, Aleksandrs Belovs, Oded Regev, and Ronald de Wolf. Efficient quantum algorithms for (gapped) group testing and junta testing. In *Proceedings of the 27th annual Symposium on Discrete Algorithms (SODA)*, pages 903–922, 2016. DOI: 10.1137/1.9781611974331.ch65.

[ACL11] Andris Ambainis, Andrew M Childs, and Yi-Kai Liu. Quantum property testing for bounded-degree graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 6845, pages 365–376. Springer Berlin Heidelberg, 2011. DOI: 10.1007/978-3-642-22935-0_31.

[ADL$^+$94] Noga Alon, Richard A Duke, Hanno Lefmann, Vojtech Rodl, and Raphael Yuster. The algorithmic aspects of the regularity lemma. *Journal of Algorithms*, 16(1):80–109, 1994. DOI: 10.1006/jagm.1994.1005.

[ADW22] Simon Apers and Ronald De Wolf. Quantum speedup for graph sparsification, cut approximation, and Laplacian solving. *SIAM Journal on Computing*, 51(6):1703–1742, 2022. DOI: 10.1137/21M1391018.

[AGSS23] Simon Apers, Sander Gribling, Sayantan Sen, and Dániel Szabó. A (simple) classical algorithm for estimating Betti numbers. *Quantum*, 7:1202, 2023. DOI: 10.22331/q-2023-12-06-1202.

[AKN98] Dorit Aharonov, Alexei Kitaev, and Noam Nisan. Quantum circuits with mixed states. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, page 20–30. Association for Computing Machinery, 1998. DOI: 10.1145/276698.276708.

[Amb04] A. Ambainis. Quantum search algorithms. *SIGACT News*, 35(2):22–35, 2004. DOI: 10.1145/992287.992296.

[Amb05] Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory of Computing*, 1(3):37–46, 2005. DOI: 10.4086/toc.2005.v001a003.

[AMS24] Bernardo Ameneyro, Vasileios Maroulas, and George Siopsis. Quantum persistent homology. *Journal of Applied and Computational Topology*, 8(7):1961–1980, 2024. DOI: 10.1007/s41468-023-00160-7.

[AMSS25] Simon Apers, Frédéric Magniez, Sayantan Sen, and Dániel Szabó. Quantum property testing in sparse directed graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 353, pages 32:1–32:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2025.32.

[AS04] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, 2004. DOI: 10.1145/1008731.1008735.

[AS05] Noga Alon and Asaf Shapira. Every monotone graph property is testable. In *Proceedings of the 37th annual ACM Symposium on Theory of Computing (STOC)*, pages 128–137. Association for Computing Machinery, 2005. DOI: 10.1145/1060590.1060611.

[AS08] Noga Alon and Asaf Shapira. A characterization of the (natural) graph properties testable with one-sided error. *SIAM Journal on Computing*, 37(6):1703–1727, 2008. DOI: 10.1137/06064888X.

[AS19] Simon Apers and Alain Sarlette. Quantum fast-forwarding: Markov chains and graph property testing. *Quantum Information & Computation*, 19(3–4):181–213, 2019. DOI: 10.5555/3370245.3370246.

[AUC$^+$24] Ismail Yunus Akhalwaya, Shashanka Ubaru, Kenneth L. Clarkson, Mark S. Squillante, Vishnu Jejjala, Yang-Hui He, Kugendran Naidoo, Vasileios Kalantzis, and Lior Horesh. Topological data analysis on noisy quantum computers. In *The 12th International Conference on Learning Representations (ICLR)*, 2024.

[BAD21] Anuraag Bukkuri, Noemi Andor, and Isabel K. Darcy. Applications of topological data analysis in oncology. *Frontiers in Artificial Intelligence*, 4, 2021. DOI: 10.3389/frai.2021.659037.

[Bar79] J. A. Barker. A quantum-statistical Monte Carlo method; path integrals with boundary conditions. *The Journal of Chemical Physics*, 70(6):2914–2918, 1979. DOI: 10.1063/1.437829.

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. DOI: 10.1145/502090.502097.

[BBHT99] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. *Tight Bounds on Quantum Searching*, chapter 10, pages 187–199. John Wiley & Sons, Ltd, 1999. DOI: 10.1002/3527603093.ch10.

[BDCG+20] Shalev Ben-David, Andrew M Childs, András Gilyén, William Kretschmer, Supartha Podder, and Daochen Wang. Symmetries, graph properties, and quantum speedups. In *61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 649–660. IEEE, 2020. DOI: 10.1109/FOCS46700.2020.00066.

[Ben80] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22(5):563–591, 1980. DOI: 10.1007/BF01011339.

[BFNR08] Harry Buhrman, Lance Fortnow, Ilan Newman, and Hein Röhrig. Quantum property testing. *SIAM Journal on Computing*, 37(5):1387–1400, 2008. DOI: 10.1137/S0097539704442416.

[BHH11] Sergey Bravyi, Aram W. Harrow, and Avinatan Hassidim. Quantum algorithms for testing properties of distributions. *IEEE Transactions on Information Theory*, 57(6):3971–3981, 2011. DOI: 10.1109/TIT.2011.2134250.

[BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Information*, page 53–74, 2002. DOI: 10.1090/conm/305/05215.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *Proceedings of the 3rd Latin American Symposium on Theoretical Informatics (LATIN)*, pages 163–169. Springer Berlin Heidelberg, 1998. DOI: 10.1007/BFb0054319.

[BK89] Manuel Blum and Sampath Kannan. Designing programs that check their work. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 86–97. Association for Computing Machinery, 1989. DOI: 10.1145/73007.73015.

[BKT20] Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: Tight quantum query bounds via dual polynomials. *Theory of Computing*, 16(10):1–71, 2020. DOI: 10.4086/toc.2020.v016a010.

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–83. Association for Computing Machinery, 1990. DOI: 10.1145/100216.100225.

[BOT02] Andrej Bogdanov, Kenji Obata, and Luca Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *43rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 93–102. IEEE Computer Society, 2002. DOI: 10.1109/SFCS.2002.1181886.

[BR02] Michael A Bender and Dana Ron. Testing properties of directed graphs: acyclicity and connectivity. *Random Structures & Algorithms*, 20(2):184–205, 2002. DOI: 10.1002/rsa.10023.

[BSG+24] Dominic W. Berry, Yuan Su, Casper Gyurik, Robbie King, Joao Basso, Alexander Del Toro Barba, Abhishek Rajput, Nathan Wiebe, Vedran Dunjko, and Ryan Babbush. Analyzing prospects for quantum advantage in topological data analysis. *PRX Quantum*, 5:010319, 2024. DOI: 10.1103/PRXQuantum.5.010319.

[BT20] Mark Bun and Justin Thaler. A nearly optimal lower bound on the approximate degree of AC^0. *SIAM Journal on Computing*, 49(4):FOCS17–59–FOCS17–96, 2020. DOI: 10.1137/17M1161737.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. DOI: 10.1137/S0097539796300921.

[BY22] Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing - Problems and Techniques*. Springer Singapore, 1 edition, 2022. DOI: 10.1007/978-981-16-8622-1.

[Car09] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009. DOI: 10.1090/S0273-0979-09-01249-X.

[CC24] Chris Cade and P. Marcos Crichigno. Complexity of supersymmetric systems and the cohomology problem. *Quantum*, 8:1325, 2024. DOI: 10.22331/q-2024-04-30-1325.

[CK24] Marcos Crichigno and Tamara Kohler. Clique homology is QMA1-hard. *Nature Communications*, 15(1):9846, 2024. DOI: 10.1038/s41467-024-54118-z.

[CL87] Raul Cordovil and Bernt Lindström. Simplicial matroids. In *Combinatorial Geometries*. Cambridge University Press, 1987. DOI: 10.1017/CBO9781107325715.

[CNPS17] André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *Advances in Cryptology (ASIACRYPT)*, pages 211–240. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-70697-9_8.

[Cor78] Raul Cordovil. Sur les géometries simpliciales. *Comptes Rendus hebdomadaires des séances de l'Académie des Sciences*, 286(25):1219–1222, 1978.

[CPS16] Artur Czumaj, Pan Peng, and Christian Sohler. Relating two property testing models for bounded degree directed graphs. In *Proceedings of the 48th annual Symposium on Theory of Computing (STOC)*, pages 1033–1045. Association for Computing Machinery, 2016. DOI: 10.1145/2897518.2897575.

[CR70] Henry H. Crapo and Gian-Carlo Rota. On the foundations of combinatorial theory ii. combinatorial geometries. *Studies in Applied Mathematics*, 49(2):109–133, 1970. DOI: 10.1002/sapm1970492109.

[CS10] Artur Czumaj and Christian Sohler. Sublinear-time algorithms. In Oded Goldreich, editor, *Property Testing: Current Research and Surveys*, pages 41–64. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-16367-8_5.

[CWS+25] Santiago Cifuentes, Samson Wang, Thais L. Silva, Mario Berta, and Leandro Aolita. Quantum computational complexity of matrix functions. *arXiv:2410.13937*, 2025. DOI: 10.48550/arXiv.2410.13937.

[CY19] Hubie Chen and Yuichi Yoshida. Testability of homomorphism inadmissibility: Property testing meets database theory. In *Proceedings of the 38th*

ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS), pages 365–382. Association for Computing Machinery, 2019. DOI: 10.1145/3294052.3319679.

[Deu85] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London*, 400(1818):97–117, 1985. DOI: 10.1098/rspa.1985.0070.

[dGdW02] Mart de Graaf and Ronald de Wolf. On quantum versions of the Yao principle. In *19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 347–358. Springer Berlin Heidelberg, 2002. DOI: 10.1007/3-540-45841-7_28.

[DH96] Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *arXiv:9607014*, 1996. DOI: 10.48550/arXiv.quant-ph/9607014.

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London*, 439(1907):553–558, 1992. DOI: 10.1098/rspa.1992.0167.

[DP09] Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511581274.

[DR02] Art Duval and Victor Reiner. Shifted simplicial complexes are Laplacian integral. *Transactions of the American Mathematical Society*, 354(11):4313–4344, 2002. DOI: 10.1090/S0002-9947-02-03082-9.

[DSTS17] Dean Doron, Amir Sarid, and Amnon Ta-Shma. On approximating the eigenvalues of stochastic matrices in probabilistic logspace. *computational complexity*, 26:393–420, 2017. DOI: 10.1007/s00037-016-0150-y.

[EKK$^+$98] Funda Ergün, Sampath Kannan, S Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 259–268. Association for Computing Machinery, 1998. DOI: 10.1145/276698.276757.

[Ele10] Gábor Elek. *Betti numbers are testable*, pages 139–149. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-13580-4_6.

[ELZ02] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002. DOI: 10.1007/s00454-002-2885-2.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, 1984. DOI: 10.1016/S0019-9958(84)80056-X.

[Fey82] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982. DOI: 10.1007/BF02650179.

[Fis01] Eldar Fischer. The art of uninformed decisions. *Bulletin of the EATCS*, 75:97, 2001.

[FL50] George E Forsythe and Richard A Leibler. Matrix inversion by a Monte Carlo method. *Mathematics of Computation*, 4(31):127–129, 1950. DOI: 10.1090/S0025-5718-1950-0038138-X.

[FN07] Eldar Fischer and Ilan Newman. Testing versus estimation of graph properties. *SIAM Journal on Computing*, 37(2):482–501, 05 2007. DOI: 10.1137/060652324.

[FNY$^+$20] Sebastian Forster, Danupon Nanongkai, Liu Yang, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Computing and testing small connectivity in near-linear time and queries via fast local cut algorithms. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2046–2065. Society for Industrial and Applied Mathematics, 2020. DOI: 10.5555/3381089.3381215.

[Fox11] Jacob Fox. A new proof of the graph removal lemma. *Annals of Mathematics*, 174(1):561–579, 2011. DOI: 10.4007/annals.2011.174.1.17.

[Fri98] Joel Friedman. Computing Betti numbers via combinatorial Laplacians. *Algorithmica*, 21(4):331–346, 1998. DOI: 10.1007/PL00009218.

[Für95] Zoltán Füredi. Extremal hypergraphs and combinatorial geometry. In *Proceedings of the International Congress of Mathematicians*, pages 1343–1352. Birkhäuser Basel, 1995. DOI: 10.1007/978-3-0348-9078-6_129.

[GCD22] Casper Gyurik, Chris Cade, and Vedran Dunjko. Towards quantum advantage via topological data analysis. *Quantum*, 6:855, 2022. DOI: 10.22331/q-2022-11-10-855.

[GGR98] Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998. DOI: 10.1145/285055.285060.

[GKNR12] Oded Goldreich, Michael Krivelevich, Ilan Newman, and Eyal Rozenberg. Hierarchy theorems for property testing. *computational complexity*, 21(1):129–192, 2012. DOI: 10.1007/s00037-011-0022-4.

[GKS23] Lior Gishboliner, Nick Kushnir, and Asaf Shapira. Testing Versus Estimation of Graph Properties, Revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 275, pages 46:1–46:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2023.46.

[GLG22] Sevag Gharibian and François Le Gall. Dequantizing the quantum singular value transformation: Hardness and applications to quantum chemistry and the quantum PCP conjecture. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 19–32. Association for Computing Machinery, 2022. DOI: 10.1145/3519935.3519991.

[Gol02] Timothy E. Goldberg. Combinatorial Laplacians of simplicial complexes. Senior Project, Bard College, 2002. Link.

[Gol10] Oded Goldreich. *Introduction to testing graph properties*, pages 105–141. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-16367-8_7.

[Gol17] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[Gol25] Oded Goldreich. *On Testing Hamiltonicity in the Bounded Degree Graph Model*, pages 282–292. Springer Nature Switzerland, 2025. DOI: 10.1007/978-3-031-88946-2_15.

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. DOI: 10.1007/s00453-001-0078-7.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, page 212–219. Association for Computing Machinery, 1996. DOI: 10.1145/237814.237866.

[Hat02] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[Hay22] Ryu Hayakawa. Quantum algorithm for persistent Betti numbers and topological data analysis. *Quantum*, 6:873, 2022. DOI: 10.22331/q-2022-12-07-873.

[HLM17] Aram W Harrow, Cedric Yen-Yu Lin, and Ashley Montanaro. Sequential measurements, disturbance and property testing. In *Proceedings of the 28th Annual Symposium on Discrete Algorithms (SODA)*, pages 1598–1611. Society for Industrial and Applied Mathematics, 2017. DOI: 10.5555/3039686.3039791.

[Hod41] W. V. D. Hodge. *The Theory and Applications of Harmonic Integrals*. Cambridge University Press, 1941.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. DOI: 10.1080/01621459.1963.10500830.

[HS12] Frank Hellweg and Christian Sohler. Property testing in sparse directed graphs: Strong connectivity and subgraph-freeness. In *20th Annual European Symposium on Algorithms (ESA)*, pages 599–610. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-33090-2_52.

[Jon] Jakob Jonsson. Introduction to simplicial homology. Royal Institute of Technology, Stockholm. Link.

[JR23] Samuel Jaques and Arthur G. Rattew. Qram: A survey and critique. *arXiv:2305.10310*, 2023. DOI: 10.48550/arXiv.2305.10310.

[KMH$^+$21] Aditi S Krishnapriyan, Joseph Montoya, Maciej Haranczyk, Jens Hummelshøj, and Dmitriy Morozov. Machine learning with persistent homology and chemical word embeddings improves prediction accuracy and interpretability in metal-organic frameworks. *Scientific Reports*, 11(1):8888, 2021. DOI: 10.1038/s41598-021-88027-8.

[Kut05] Samuel Kutin. Quantum lower bound for the collision problem with small range. *Theory of Computing*, 1(2):29–36, 2005. DOI: 10.4086/toc.2005.v001a002.

[Lee09] Troy Lee. A note on the sign degree of formulas. *arXiv:0909.4607*, 2009. DOI: 10.48550/arXiv.0909.4607.

[LGZ16] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of data. *Nature Communications*, 7(1):10138, 2016. DOI: 10.1038/ncomms10138.

[LZ19] Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. In *Advances in Cryptology (EUROCRYPT)*, pages 189–218. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-17659-4_7.

[Man80] Yuri I. Manin. *Computable and Non-Computable*. Sovetskoe Radio, 1980.

[MdW16] Ashley Montanaro and Ronald de Wolf. *A Survey of Quantum Property Testing*. Number 7 in Graduate Surveys. Theory of Computing Library, 2016. DOI: 10.4086/toc.gs.2016.007.

[MGB22] Sam McArdle, András Gilyén, and Mario Berta. A streamlined quantum algorithm for topological data analysis with exponentially fewer qubits. *arXiv:2209.12887*, 2022. DOI: 10.48550/arXiv.2209.12887.

[MTZ20] Nikhil S. Mande, Justin Thaler, and Shuchen Zhu. Improved approximate degree bounds for k-distinctness. In *Proceedings of the 15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*, volume 158, pages 2:1–2:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. DOI: 10.4230/LIPIcs.TQC.2020.2.

[MWW22] Facundo Mémoli, Zhengchao Wan, and Yusu Wang. Persistent Laplacians: Properties, algorithms and implications. *SIAM Journal on Mathematics of Data Science*, 4(2):858–884, 2022. DOI: 10.1137/21M1435471.

[Nan] Vidit Nanda. Computational algebraic topology lecture notes. University of Oxford. Link.

[New18] J. Andrew Newman. *Torsion in Homology of Random Simplicial Complexes*. PhD thesis, The Ohio State University, 2018.

[OR11] Yaron Orenstein and Dana Ron. Testing Eulerianity and connectivity in directed sparse graphs. *Theoretical Computer Science*, 412(45):6390–6408, 2011. DOI: 10.1016/j.tcs.2011.06.038.

[PEv$^+$16] Pratyush Pranav, Herbert Edelsbrunner, Rien van de Weygaert, Gert Vegter, Michael Kerber, Bernard J. T. Jones, and Mathijs Wintraecken. The topology of the cosmic web in terms of persistent Betti numbers. *Monthly Notices of the Royal Astronomical Society*, 465(4):4281–4310, 2016. DOI: 10.1093/mnras/stw2862.

[PRR06] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006. DOI: 10.1016/j.jcss.2006.03.002.

[PW23] Pan Peng and Yuyang Wang. An optimal separation between two property testing models for bounded degree directed graphs. In *50th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 261, pages 96:1–96:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPIcs.ICALP.2023.96.

[Rie17] Bastian Rieck. *Persistent Homology in Multivariate Data Visualization*. PhD thesis, Heidelberg University, 2017. DOI: 10.11588/heidok.00022914.

[Ron10] Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5(2):73–205, 2010. DOI: 10.1561/0400000029.

[RRSS09] Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009. DOI: 10.1137/070701649.

[RS11] Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms. *SIAM Journal on Discrete Mathematics*, 25(4):1562–1588, 2011. DOI: 10.1137/100791075.

[SA25] Dániel Szabó and Simon Apers. Holey graphs: Very large Betti numbers are testable. In *SOFSEM 2025: Theory and Practice of Computer Science*, pages 298–310. Springer Nature Switzerland, 2025. DOI: 10.1007/978-3-031-82697-9_22.

[She11] Alexander A. Sherstov. The pattern matrix method. *SIAM Journal on Computing (SICOMP)*, 40(6):1969–2000, 2011. DOI: 10.1137/080733644.

[She13] Alexander A. Sherstov. The intersection of two halfspaces has high threshold degree. *SIAM Journal on Computing*, 42(6):2329–2374, 2013. DOI: 10.1137/100785260.

[Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 124–134. IEEE, 1994. DOI: 10.1109/SFCS.1994.365700.

[SL23] Alexander Schmidhuber and Seth Lloyd. Complexity-theoretic limitations on quantum algorithms for topological data analysis. *PRX Quantum*, 4:040349, 2023. DOI: 10.1103/PRXQuantum.4.040349.

[SV14] Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends in Theoretical Computer Science*, 9(2):125–210, 2014. DOI: 10.1561/0400000065.

[SZ09] Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *Quantum Information & Computation*, 9(5):444–460, 2009. DOI: 10.5555/2011791.2011798.

[Sze78] Endre Szemerédi. Regular partitions of graphs. In *Problèmes combinatoires et théorie des graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976)*, volume 260 of *Colloq. Internat. CNRS*, pages 399–401. CNRS, Paris, 1978.

[Tur37] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. DOI: 10.1112/plms/s2-42.1.230.

[vAGN24] Joran van Apeldoorn, Sander Gribling, and Harold Nieuwboer. Basic quantum subroutines: finding multiple marked elements and summing numbers. *Quantum*, 8:1284, 2024. DOI: 10.22331/q-2024-03-14-1284.

[WNW20] Rui Wang, Duc Duy Nguyen, and Guo-Wei Wei. Persistent spectral graph. *International Journal for Numerical Methods in Biomedical Engineering*, 36(9):e3376, 2020. DOI: 10.1002/cnm.3376.

[Yao77] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977. DOI: 10.1109/SFCS.1977.24.

[YI10a] Yuichi Yoshida and Hiro Ito. Query-number preserving reductions and linear lower bounds for testing. *IEICE transactions on Information and Systems*, E93.D(2):233–240, 2010. DOI: 10.1587/transinf.E93.D.233.

[YI10b] Yuichi Yoshida and Hiro Ito. Testing k-edge-connectivity of digraphs. *Journal of Systems Science and Complexity*, 23(1):91–101, 2010. DOI: 10.1007/s11424-010-9280-5.