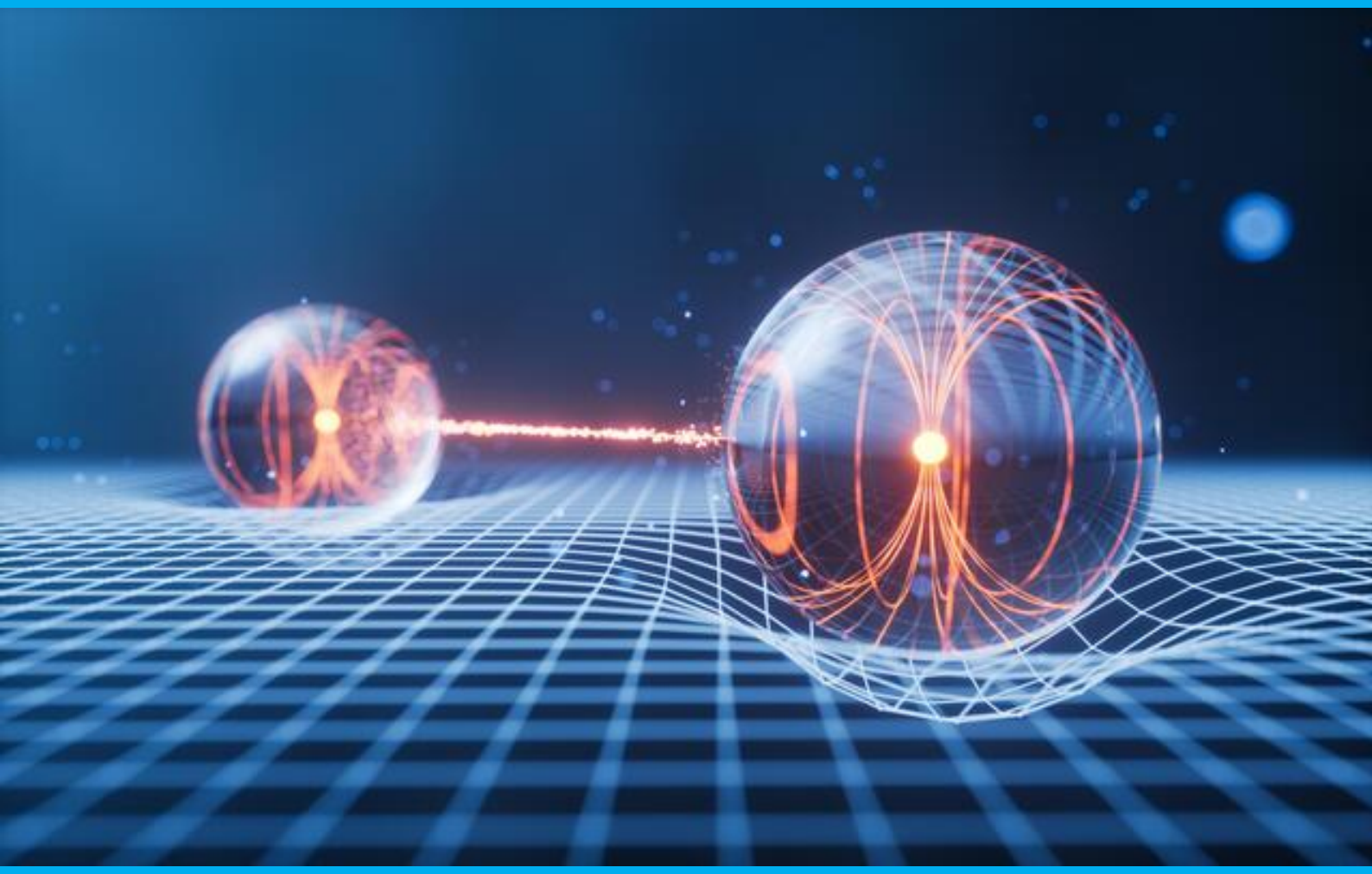
A comparison between quantum search algorithms of bosons and fermions

T.B. van Gelder



A comparison between quantum search algorithms of bosons and fermions

by

T.B. van Gelder

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on the 11th of February 2026.

Student number: 5852358
Project duration: 11-07-2025 to 11-02-2026
Thesis committee: Dr. J.L.A. Dubbeldam, Prof. Dr. Y.M. Blanter
Dr. M.P.T. Caspers, Dr. M.N. Ali

Quantum search algorithms provide a quadratic speed-up over classical search for unstructured databases. While single-particle quantum search is well understood for a large set of graphs, considerably less is known about the behavior of multi-particle quantum search algorithms. In this thesis, we consider continuous-time quantum search of multiple bosons or fermions over arbitrary graphs and develop a method to identify the marked vertex after multiple measurements. We investigate the performance of bosonic search as compared to fermionic search by simulating this search algorithm and calculating the runtime numerically. It is found that, though bosonic search is substantially faster than fermionic search, the fermionic runtime may not scale as fast with the graph size N as was shown in [6], if our method of post-measurement marked vertex identification is used. Furthermore, it is shown that the optimal hopping rate for a quantum search is different for bosons and fermions, and in general not equal to the optimal single particle hopping rate of [1].

Contents

1

Introduction

Why would we be interested in quantum searches on graphs? This is because a graph represents a database, and if that database is unstructured, a quantum search can find an element in that database faster than any classical search algorithm.

Classically, if no information about the internal structure of the database is available, the marked element can only be found by sequentially checking elements, resulting in a runtime that scales linearly with the database size N. In [4] it was shown that for a complete graph, a discrete-time quantum search can achieve a quadratic speed-up in finding the marked element. Farhi and Gutmann adapted the quantum search to the continuous-time regime [3] and it was later shown that, using a single particle, this quadratic speed-up can be achieved for a large set of graphs [1]. More recently, attention has shifted to multi-particle quantum search. In [6] it was shown that for a multi-fermion quantum search over a complete graph the runtime scales with $N^{1/3}$ and for a multi-boson quantum search the runtime scales as $N^{1/4}$.

In this thesis we will consider continuous-time quantum searches of bosons or fermions over arbitrary graphs. We use the same quantum search as in [6] and develop a method for determining the marked element after measurement. We implement this search algorithm in a parallel computer program that is run on the Delftblue supercomputer. Our aim is to investigate the effect that our method for post-measurement marked element identification has on the runtime of the fermionic search. Furthermore, we assess whether the optimal single-particle hopping rate as found in [1] is also optimal for the multi-particle search. Finally, we consider how the connectivity of a graph influences the fermionic runtime as compared to the bosonic runtime. Taking into account these results, we will better be able to compare bosonic with fermionic quantum search.

2

Theory

Before we can start the analysis of quantum searches over graphs, we need to understand what graphs are and how quantum systems behave. The goal of the following chapter is to introduce the graphs used in this thesis and explain the quantum notation that is used.

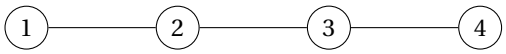
2.1. Graphs

2.1.1. Graph theory

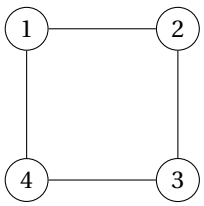
A **graph** is an abstract representation of a network. It consists of a set of **vertices** V and a set of **edges** E between these vertices. Together, these two sets form the graph $G = (V, E)$. In this thesis, we will only consider finite graphs with N vertices and we choose to name our vertices 1, 2, ... up to N, so that $V = \{1, 2, \ldots, N\}$. If vertices $i, j \in V$ are connected by an edge in E, then we write $(i, j) \in E$.

2.1.2. Deterministic graphs

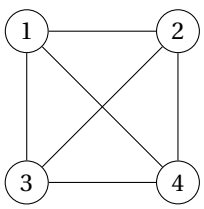
A deterministic graph has a fixed structure. If we follow the rules to build it twice, we will get the exact same graph both times. In this thesis we will use three types of deterministic graphs: line graphs, cycle graphs and complete graphs. A **line graph** L_N of N vertices is a graph where the edges form a path from vertex 1 to N. For example, L_4 looks like



A **cycle graph** C_N of N vertices is a graph where the edges form a ring over the vertices 1, 2, ..., N, 1 in that order. C_4 looks like



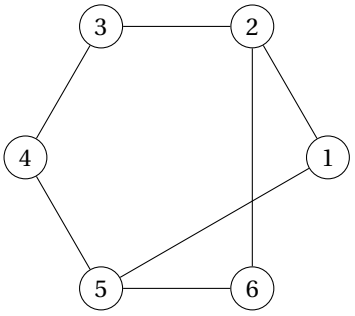
A **complete graph** K_N of N vertices is a graph where every vertex is connected to every other vertex via an edge. K_4 looks like



2.1.3. Random graphs

A random graph is generated by a stochastic process. If we generate a random graph twice using the same rules, the resulting graphs will generally be different. In this thesis we will use one type of random graph, namely the Erdős–Rényi random graph.

Erdős–Rényi random graphs The Erdős–Rényi (ER) random graph is denoted as $G(N,p)$. As before N is the number of vertices in the graph. What makes this graph random is that each possible edge in $G(N,p)$ has a probability p of being included in the graph. An instance of $G(6,\frac{1}{2})$ is shown below.



The graph $G(N,p)$ could in theory be any graph of N vertices, as long as $p \neq 0$ and $p \neq 1$. However, usually a larger p means more edges in the graph, so that each vertex is, on average, connected to more other vertices.

A graph is said to be **connected**, if for any two vertices $i, j \in V$ there exists a path (along edges in the graph) from i to j. In order to implement a quantum search algorithm on a graph, it is important that this graph is connected. For an ER random graph, according to [8], the graph is connected with large probability for p such that

$$p > \frac{\log(N)}{N} \tag{2.1}$$

2.2. Single-particle quantum systems

In order to understand the dynamics of a quantum search algorithm over a graph, one has to learn the language of quantum mechanics. Specifically, the notation of states and operators. To this end, we start with the simplest quantum system treated in this thesis, a single particle on a graph, and define a way to describe it in bra-ket notation.

2.2.1. The state of a single particle on a graph

Let us consider an arbitrary graph $G = (V,E)$ with V again the set $\{1,2,\ldots,N\}$. Classically, a particle in this graph has to be at exactly one vertex at each moment in time. Say at one moment it is at vertex $i \in V$, then the state of this quantum system at that time is $|i\rangle$. In quantum mechanics a particle can be at multiple positions at the same time, meaning that only when we *measure* the system, the particle takes on a definite position. Before measurement the particle is in a **quantum superposition** of states, and there is no way to find out in which state it actually is. In fact, we do not even talk about the particle being in a specific state before measurement. It is in all states of the superposition at once. A superposition of the particle being in both vertex i and j, we can write as

$$|\psi\rangle = \frac{\sqrt{3}}{2}|i\rangle + \frac{1}{2}|j\rangle \tag{2.2}$$

Here $|\psi\rangle$ is the state of the system, and we can express it as a superposition of the **quantum basis states** $|i\rangle$ and $|j\rangle$.

Importantly, a general quantum state $|\psi\rangle$ has to satisfy the following condition,

$$\sum_{i=1}^{N} |\langle\psi|i\rangle|^2 = 1 \tag{2.3}$$

so that it has probability 1 to be anywhere on the graph.

2.2.2. Single-particle operators

A **quantum operator** acts on a quantum state and changes it. Quantum operators can be used to calculate certain characteristics of a system, or to manipulate the state in a particular way. To illustrate these possibilities, we consider the infinite line graph with vertices $\ldots,-1,0,1,2,\ldots$ given below.

Projector operators Suppose a single particle is in a superposition $|\psi\rangle$ over this graph, and we want to determine the probability that, upon measurement, we will find it on an even vertex, that is on vertex 0 or ± 2 or ± 4 or .... This can be done using an operator: the projector operator. Let us write out the arbitrary state $|\psi\rangle$ of the system as a superposition of the basis states $|i\rangle$ for $i = \ldots,-1,0,1,\ldots$ as defined in section 2.2.1 ($|i\rangle$ being the state of one particle localized at vertex i),

$$|\psi\rangle = \cdots + \alpha_{-1}|-1\rangle + \alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle + \ldots \tag{2.4}$$

where $\alpha_i \in \mathbb{C}$. An operator is defined by how it acts on the basis states of a system (here $|i\rangle$, $i \in \mathbb{Z}$). In the case of a *projector operator* acting on a superposition, it removes certain terms and keeps the rest. Say we define our projector operator as follows.

$$\begin{cases} \hat{P}|i\rangle = |i\rangle & \text{if } i \text{ is even} \\ \hat{P}|i\rangle = 0 & \text{if } i \text{ is odd} \end{cases} \tag{2.5}$$

Then $\hat{P}$ removes all odd terms and keeps all even terms. Note that the basis states $|i\rangle$ are the eigenvectors of the projector operator $\hat{P}$, with degenerate eigenvalues 0 or 1. The eigenvalue 0 has eigenvectors $\ldots,|-1\rangle,|1\rangle,|3\rangle,\ldots$ and the eigenvalue 1 has eigenvectors $\ldots,|-2\rangle,|0\rangle,|2\rangle,\ldots$.

Why is defining a projector operator in this way beneficial? Well, if we let $\hat{P}$ act on $|\psi\rangle$, we now get

$$\hat{P}|\psi\rangle = \cdots + \alpha_{-2}|-2\rangle + \alpha_0|0\rangle + \alpha_2|2\rangle + \ldots \tag{2.6}$$

Then,

$$\begin{aligned} \langle\psi|\hat{P}|\psi\rangle &= \Big(\cdots + \alpha_{-1}^*\langle -1| + \alpha_0^*\langle 0| + \alpha_1^*\langle 1| + \ldots\Big)\Big(\cdots + \alpha_{-2}|-2\rangle + \alpha_0|0\rangle + \alpha_2|2\rangle + \ldots\Big) \\ &= \cdots + |\alpha_{-2}|^2 + |\alpha_0|^2 + |\alpha_2|^2 + \ldots \\ &= \cdots + P\Big(\text{particle ends up at vertex -2}\Big) + P\Big(\text{" at vertex 0}\Big) + P\Big(\text{" at vertex 2}\Big) + \ldots \end{aligned}$$

which equals the probability that the particle ends up at an even vertex. In this way a projector operator can be used to calculate the probability that the system will be in some arbitrary sub-state after measurement.

Step operators Now that we have seen how an operator can be used to calculate characteristics of a quantum system, let us look at how an operator can manipulate states. We are interested in an operator that lets a particle *move* one vertex to the right. Such an operator is

$$\hat{S} = \sum_{j=-\infty}^{\infty} |j+1\rangle\langle j|, \tag{2.7}$$

the step operator. Indeed, when $\hat{S}$ acts on $|i\rangle$, we get

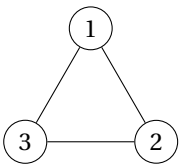
$$\hat{S}|i\rangle = \Big(\sum_{j=-\infty}^{\infty} |j+1\rangle\langle j|\Big)|i\rangle = |i+1\rangle\langle i|i\rangle = |i+1\rangle \tag{2.8}$$

In this manner an operator can be used to manipulate a quantum state.

2.2.3. Vector notation of states and operators

In this thesis we will only consider quantum states from a finite dimensional Hilbert space. Furthermore, all operators used are linear. In this case, it is convenient to write our quantum states as vectors and our operators as matrices.

To illustrate how a quantum state can be written as a vector and how an operator can be written as a matrix, we consider the complete graph K_3 as illustrated below.



A general state $|\psi\rangle$ can be expressed in vector notation as follows.

$$|\psi\rangle = \alpha_1 |1\rangle + \alpha_2 |2\rangle + \alpha_3 |3\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{2.9}$$

The step operator $\hat{S}$ on this graph (walking clockwise) written as a matrix would be

$$\hat{S} = |2\rangle\langle 1| + |3\rangle\langle 2| + |1\rangle\langle 3| = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{2.10}$$

2.2.4. The Hamiltonian and continuous-time evolution

As suggested before, operators can be used to create dynamics in a quantum system. As an example, a step operator was introduced, manipulating a state in such a way that it lets the particle 'move' from one vertex to the other. To simulate the behavior of a particle 'walking' over a graph, we could start with an initial state $|\psi_0\rangle$ and apply our step operator on this state over and over again, so that $|\psi_1\rangle = \hat{S}|\psi_0\rangle$, $|\psi_2\rangle = \hat{S}|\psi_1\rangle$, and so on. In this way, our state *evolves* over time in discrete steps under the influence of $\hat{S}$: it is called discrete-time evolution.

In reality, however, a (closed) quantum system is governed by the Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} |\psi(t)\rangle = \hat{H} |\psi(t)\rangle \tag{2.11}$$

Here $|\psi(t)\rangle$ is the state after time t and $\hat{H}$ is the **Hamiltonian** of the system. The Hamiltonian is the operator that determines the dynamics of the system. In the case of a particle walking over a graph, this could be our step operator $\hat{S}$. Note that, in this case, $\hat{H}$ is independent of t, so that we can solve the Schrödinger equation.

$$|\psi(t)\rangle = e^{-i\hat{H}t/\hbar} |\psi(0)\rangle \tag{2.12}$$

Here $|\psi(0)\rangle$ is the state of the system at $t = 0$: the initial state.

Here $\hbar = 1.054 \cdot 10^{-34}$ J·s. In our simulation, we work with a time unit $\tilde{\text{s}}$ and an energy unit $\tilde{\text{J}}$ so that $\hbar = 1\ \tilde{\text{J}}\cdot\tilde{\text{s}}$. Then (2.12) becomes,

$$|\psi(t)\rangle = e^{-i\hat{H}t} |\psi(0)\rangle \tag{2.13}$$

.

We choose to use Joules J as our energy unit, so $1\ \tilde{\text{J}} = 1$ J. Then our unit of time is given by $1\ \tilde{\text{s}} = 1.054 \cdot 10^{-34}$ s, to ensure $\hbar = 1\ \tilde{\text{J}}\cdot\tilde{\text{s}}$. Therefore, to convert from the time in our simulation to the time in seconds, one can use $t[\text{s}] = \frac{t[\tilde{\text{s}}]}{1.054 \cdot 10^{-34}}$.

Using (2.13), one can calculate how a quantum state evolves under the influence of a Hamiltonian in *continuous time*. This is how a real-world quantum system evolves and how we will simulate our quantum system.

2.3. Multi-particle quantum systems

Quantum systems typically contain more than one particle. In this thesis we will eventually work with multi-particle quantum systems. Therefore, we need to investigate the way in which multiple particles change the dynamics of a system and what notation is used.

2.3.1. Distinguishable and indistinguishable particles

When talking about multiple particles in quantum mechanics, it is important to specify whether these particles are *distinguishable* or *indistinguishable*. The state of a system of two particles we can write as $\Psi(x_1, x_2)$ (assuming the particles have no spin), meaning particle 1 is at location x_1 and particle 2 at x_2. If the particles are indistinguishable, interchanging them should not alter the probabilities of measuring any characteristic of the system, so

$$P(x_1, x_2) = P(x_2, x_1). \tag{2.14}$$

Since the probability is proportional to the wavefunction squared, we have

$$|\Psi(x_1, x_2)|^2 = |\Psi(x_2, x_1)|^2 \Rightarrow \Psi(x_1, x_2) = c\Psi(x_2, x_1), \tag{2.15}$$

where c can represent any complex number with norm 1. In 3D, swapping the particles twice should result in the same state as we started with [7]. Thus $c^2\Psi(x_1, x_2) = \Psi(x_1, x_2)$, so that $c = \pm 1$. If $c = 1$, then $\Psi(x_1, x_2) = +\Psi(x_2, x_1)$ and we call the state **bosonic**. If $c = -1$, then $\Psi(x_1, x_2) = -\Psi(x_2, x_1)$ and we call the state **fermionic**[*].

The Pauli exclusion principle Suppose two fermions occupy the same position x_1. Then $\Psi(x_1, x_1) = -\Psi(x_1, x_1)$, so $\Psi(x_1, x_1) = 0$: this state is not allowed. In general, no two fermions may occupy the same state. This is called the Pauli exclusion principle.

From now on, when we consider multi-particle systems, we talk about multiple *indistinguishable particles*. Always all particles will be either bosons or fermions.

2.3.2. The Fock basis

In a system of M bosons or fermions a simpler notation is used than the one in 2.3.1. It is called the **Fock basis**. Instead of writing the location of each particle, we write the number of particles on each vertex. In a graph of four vertices 1, 2, 3 and 4, and three bosons, we can write the state $|1020\rangle$, meaning vertex 1 contains one boson and vertex 3 contains two. (Note again that the order is important here.). Any state written in this way satisfies $\Psi(x_1, x_2) = +\Psi(x_2, x_1)$ by definition ($\Psi(x_1, x_2) = -\Psi(x_2, x_1)$ for fermions). A general Fock state we can write as $|n_1 n_2 \dots n_N\rangle$, where n_i is the number of bosons or fermions at vertex i.

2.3.3. Operators for multiple particles

As mentioned earlier, a single-particle operator is defined by how it acts on the basis states $|i\rangle$. Likewise, a multi-particle operator is defined by how it acts on the Fock states $|n_1 \dots n_N\rangle$. For example, an operator $\hat{N}_{\text{total}}$ counting the total number of particles is defined by

$$\hat{N}_{\text{total}} |n_1 \dots n_N\rangle = (n_1 + \dots + n_N) |n_1 \dots n_N\rangle \tag{2.16}$$

One can construct operators working on multiple vertices, like the one above, from operators working on single vertices. Consider for example the number operator $\hat{n}$ working on a single vertex

$$\hat{n} |n\rangle = n |n\rangle \tag{2.17}$$

[*]If we consider two particles with spin, the total state of the system can be written as the product of its spatial and spin components: $\Psi(x_1, s_1, x_2, s_2) = \psi(x_1, x_2) \otimes \chi(s_1, s_2)$. In this case, the exchange symmetry requirement (bosonic or fermionic) applies to the total wavefunction Ψ. Consequently, a fermionic state may have a symmetric spatial wavefunction provided that the spin state is antisymmetric.

Here $|n\rangle$ is the state of n particles on our vertex. Then, if we want to apply this operator on the Fock state, we use that a Fock state of N vertices is in fact a tensor product of N single-vertex states

$$|n_1 n_2 \dots n_N\rangle = |n_1\rangle \otimes |n_2\rangle \otimes \cdots \otimes |n_N\rangle \tag{2.18}$$

Thus, the operator $\hat{N}_i$ that applies $\hat{n}$ on vertex i, is $\hat{N}_i = I \otimes \cdots \otimes \underbrace{\hat{n}}_{\text{at vertex } i} \otimes \cdots \otimes I$, as

$$\begin{aligned}\hat{N}_i |n_1 \dots n_N\rangle &= \left(I \otimes \cdots \otimes \underbrace{\hat{n}}_{\text{at vertex } i} \otimes \cdots \otimes I\right) |n_1\rangle \otimes \cdots \otimes |n_N\rangle \\ &= |n_1\rangle \otimes \cdots \otimes \underbrace{\hat{n}\,|n_i\rangle}_{\text{at vertex } i} \otimes \cdots \otimes |n_N\rangle \\ &= n_i\, |n_1\rangle \otimes \cdots \otimes |n_N\rangle\end{aligned}$$

Bosonic creation and annihilation operators Two important examples of multi-particle operators used in this thesis are the creation and annihilation operators. The bosonic annihilation operator $\hat{b}_i$ destroys or deletes a boson at vertex i.

$$\hat{b}_i\, |n_1 \dots n_i \dots n_N\rangle = \sqrt{n_i}\, |n_1 \dots n_i - 1 \dots n_N\rangle \quad \text{whenever } n_i \geq 1 \tag{2.19}$$

If $n_i = 0$, then acting $\hat{b}_i$ on the state results in 0. The bosonic creation operator is the Hermitian conjugate $\hat{b}_i^\dagger$ of the bosonic annihilation operator. It creates or adds a boson at vertex i.

$$\hat{b}_i^\dagger\, |n_1 \dots n_i \dots n_N\rangle = \sqrt{n_i + 1}\, |n_1 \dots n_i + 1 \dots n_N\rangle \tag{2.20}$$

$\hat{b}_i$ and $\hat{b}_i^\dagger$ have to satisfy the so-called canonical commutation relations (CCR).

$$\begin{cases} [\hat{b}_i, \hat{b}_j^\dagger] = \hat{b}_i \hat{b}_j^\dagger - \hat{b}_j^\dagger \hat{b}_i = \delta_{ij} \\ [\hat{b}_i, \hat{b}_j] = \hat{b}_i \hat{b}_j - \hat{b}_j \hat{b}_i = 0 \\ [\hat{b}_i^\dagger, \hat{b}_j^\dagger] = \hat{b}_i^\dagger \hat{b}_j^\dagger - \hat{b}_j^\dagger \hat{b}_i^\dagger = 0 \end{cases} \tag{2.21}$$

These relations make sure that when $\hat{b}_i$ or $\hat{b}_i^\dagger$ acts on a bosonic state, the resulting state is bosonic as well. Any operator used in a bosonic system has to satisfy the commutation relations.

Fermionic creation and annihilation operators The fermionic annihilation operator $\hat{f}_i$ and creation operator $\hat{f}_i^\dagger$ are the fermionic counterparts of the bosonic annihilation and creation operators: they respectively destroy and create a fermion at site i.

$$\begin{aligned}\hat{f}_i\, |n_1 \dots \underbrace{1}_{\text{vertex } i} \dots n_N\rangle &= |n_1 \dots \underbrace{0}_{\text{vertex } i} \dots n_N\rangle \\ \hat{f}_i^\dagger\, |n_1 \dots \underbrace{0}_{\text{vertex } i} \dots n_N\rangle &= |n_1 \dots \underbrace{1}_{\text{vertex } i} \dots n_N\rangle\end{aligned}$$

$\hat{f}_i$ and $\hat{f}_i^\dagger$ have to satisfy the so-called canonical anti-commutation relations (CAR).

$$\begin{cases} \{\hat{f}_i, \hat{f}_j^\dagger\} = \hat{f}_i \hat{f}_j^\dagger + \hat{f}_j^\dagger \hat{f}_i = \delta_{ij} \\ \{\hat{f}_i, \hat{f}_j\} = \hat{f}_i \hat{f}_j + \hat{f}_j \hat{f}_i = 0 \\ \{\hat{f}_i^\dagger, \hat{f}_j^\dagger\} = \hat{f}_i^\dagger \hat{f}_j^\dagger + \hat{f}_j^\dagger \hat{f}_i^\dagger = 0 \end{cases} \tag{2.22}$$

These relations make sure that when $\hat{f}_i$ or $\hat{f}_i^\dagger$ acts on a fermionic state, the resulting state is fermionic as well. Any operator used in a fermionic system has to satisfy the anti-commutation relations.

2.3.4. The Slater determinant

Writing a multi-particle state in the Fock basis ensures symmetry or anti-symmetry for bosonic and fermionic states respectively. A way to write fermionic states without using the Fock basis is via the **Slater determinant**. Suppose we have M single-particle states

$$|\phi_1\rangle = \phi_1(1)\,|1\rangle + \cdots + \phi_1(N)\,|N\rangle \tag{2.23}$$

$$\vdots \tag{2.24}$$

$$|\phi_M\rangle = \phi_M(1)\,|1\rangle + \cdots + \phi_M(N)\,|N\rangle\,, \tag{2.25}$$

where $\phi_k(1),\ldots,\phi_k(N) \in \mathbb{C}$ are the coefficients of the k-th particle. Each particle is normalized, so $\langle\phi_k|\phi_k\rangle = 1$ for $k = 1,2,\ldots,M$. The multi-particle state, we can write as

$$|\psi\rangle = \sum_{x_1,\ldots,x_M \in V} \Psi(x_1,\ldots,x_M)\,|x_1,\ldots,x_M\rangle\,, \tag{2.26}$$

where $|x_1,\ldots,x_M\rangle$ is the state where particle 1 is at vertex x_1, particle 2 at vertex x_2, and so on. $\Psi(x_1,\ldots,x_M) \in \mathbb{C}$ is the coefficient before each term. Since the state is fermionic, we need that

$$\Psi(\ldots,x_k,\ldots,x_l,\ldots) = -\Psi(\ldots,x_l,\ldots,x_k,\ldots) \text{ for all } k,l = 1,\ldots M. \tag{2.27}$$

We can build the multi-particle state from the single-particle states, by writing each coefficient $\Psi(x_1,\ldots,x_M)$ as a determinant: the Slater determinant.

$$\Psi(x_1,\ldots,x_M) = \frac{1}{\sqrt{M!}} \begin{vmatrix} \phi_1(x_1) & \ldots & \phi_1(x_N) \\ \vdots & \ddots & \vdots \\ \phi_M(x_1) & \ldots & \phi_M(x_N) \end{vmatrix} \tag{2.28}$$

Then,

$$\Psi(\ldots,x_k,\ldots,x_l,\ldots) = \frac{1}{\sqrt{M!}} \begin{vmatrix} \ldots & \phi_1(x_k) & \ldots & \phi_1(x_l) & \ldots \\ \ldots & \phi_2(x_k) & \ldots & \phi_2(x_l) & \ldots \\ & \vdots & & \vdots & \\ \ldots & \phi_M(x_k) & \ldots & \phi_M(x_l) & \ldots \end{vmatrix} \text{ and} \tag{2.29}$$

(2.30)

$$\Psi(\ldots,x_l,\ldots,x_k,\ldots) = \frac{1}{\sqrt{M!}} \begin{vmatrix} \ldots & \phi_1(x_l) & \ldots & \phi_1(x_k) & \ldots \\ \ldots & \phi_2(x_l) & \ldots & \phi_2(x_k) & \ldots \\ & \vdots & & \vdots & \\ \ldots & \phi_M(x_l) & \ldots & \phi_M(x_k) & \ldots \end{vmatrix}. \tag{2.31}$$

We see that switching the positions of particles k and l corresponds to switching the columns of the determinant. When the columns of a matrix are interchanged, the sign of its determinant is flipped. Therefore $\Psi(\ldots,x_k,\ldots,x_l,\ldots) = -\Psi(\ldots,x_l,\ldots,x_k,\ldots)$ is satisfied when we use the Slater determinant. The constant $1/\sqrt{M!}$ in front of the determinant ensures the resulting multi-particle state is normalized, whenever the single-particle states are orthogonal, so when $\langle\phi_k|\phi_l\rangle = 0$ for all $k \neq l$. Thus, choosing $\Psi(x_1,\ldots,x_M)$ as the Slater determinant, results in a valid fermionic state.

3

A multi-particle quantum walk as search algorithm

In the following chapter we will explain what a single-particle quantum search is and provide an analysis showing the advantage this search has over a classical search algorithm in searching an unstructured database. After this, we will develop the multi-particle quantum search model that is used in this thesis.

3.1. Quantum search in an unstructured database

3.1.1. Classical search in an unstructured database

Before starting on the quantum search, let us look at how a classical search of an unstructured database looks like. Firstly, what is an unstructured database, and why can it be represented as a graph? Suppose we have a database of N items, labeled $1, 2, \ldots, N$, and a Boolean function

$$f : \{1, 2, \ldots, N\} \to \{0, 1\} \tag{3.1}$$

such that for only one item w, we have $f(w) = 1$. This w we call the marked item, and it is the algorithm's job to find it. This database is unstructured, in the sense that there are no rules categorizing its elements, no tricks one could follow to find a certain element faster. The only way to search this database would be to manually check on each element $i \in \{1, 2, \ldots, N\}$ whether it is the marked element (whether $f(i) = 1$). In a database of size N, this will take on average $N/2$ checks, so that the time it takes the classical search to find this element scales as $\Theta(N)$. This is the fastest classical search that is possible on an unstructured database.

The database of N elements can be seen as a graph with N vertices. One of these vertices, vertex w, is the marked vertex. The edges represent the possible steps one can take in a search of this graph. If the database can be searched in any order, then this means that every vertex is connected to every other vertex, so that we have a complete graph. The analogy between databases and graphs becomes more useful when considering quantum searches.

3.1.2. Grover's algorithm

An unstructured database could also be searched using a quantum search. We represent our database as a *complete* graph, and consider a quantum system of one particle in this graph. The idea of a quantum search is to construct the Hamiltonian of this system in such a way that the particle will 'move to the marked vertex'. More precisely, we construct the Hamiltonian so that after evolving the system for a certain time, there is a large probability that the particle occupies the marked vertex. After this time, we measure the location of the particle, and this location becomes our guess for which vertex is the marked vertex. This is how a quantum search can find an element in an unstructured database.

The single-particle quantum search considered in this thesis is evolved by the following Hamiltonian.

$$\hat{H} = \underbrace{-\gamma \sum_{\substack{i,j\in V \\ i\neq j}} |i\rangle\langle j|}_{\text{walk term}} \underbrace{-|w\rangle\langle w|}_{\text{oracle term}} \tag{3.2}$$

where $\gamma > 0$ is the hopping rate: a rate that determines how easily the particle hops from vertex to vertex. This is the standard Hamiltonian used for Grover's algorithm of a single particle on a graph, used in [2]. The walk-term consists of a step operator for each edge in the complete graph, in both directions. Broadly speaking, this term makes the particle move and 'spread out' over the graph during the evolution. The oracle-term lowers the energy of states that have a large amplitude on the marked vertex. This pulls the particle towards the marked vertex (this is explained in section A). Grover's algorithm consists of evolving the system under this Hamiltonian and measuring it whenever the particle is most likely to be at w. Then, wherever we measure the particle to be, will be our guess for the marked vertex. It was found that, using this algorithm, one finds the marked vertex after a time that scales as $\Theta(\sqrt{N})$ [3], which is a considerable speed-up as compared to the classical search. In appendix A it is shown why exactly this Hamiltonian draws the particle towards the marked vertex and how it achieves the quadratic speed-up.

3.2. Multi-particle quantum search

Now that we have some intuition for the dynamics of a quantum search, we can define our model. As opposed to the single-particle quantum search of before, our search uses M particles, introducing the new dynamics of bosons and fermions. The search takes place on an arbitrary graph $G = (V, E)$ of N vertices, with adjacency matrix A.

Bosonic search The bosonic quantum search used in this thesis, following [6] has Hamiltonian

$$\hat{H}_{\text{boson}} = -\gamma_c \sum_{i,j\in V} A_{ij}\hat{b}_i^\dagger \hat{b}_j - \hat{b}_w^\dagger \hat{b}_w \tag{3.3}$$

Here the first term takes the place of the single-particle walk term. $\hat{b}_i^\dagger \hat{b}_j$ is the multi-particle equivalent of the step operator $|i\rangle\langle j|$: $\hat{b}_j$ destroys a particle at vertex j and $\hat{b}_i^\dagger$ creates one at vertex i, effectively moving the boson from j to i. The second term in this Hamiltonian replaces the single-particle oracle-term.

The initial state we use, and that is used in [6] is the state of all bosons in a uniform superposition over all vertices. The state of a single boson in a uniform superposition can be written as

$$\hat{b}_{\text{tot}}^\dagger |\text{vacuum}\rangle \text{, where } \hat{b}_{\text{tot}}^\dagger = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \hat{b}_i^\dagger, \tag{3.4}$$

and $|\text{vacuum}\rangle$ is the state with no bosons. Our initial state $|\psi(0)\rangle$ consists of all M bosons being in this superposition, and is therefore constructed by applying $\hat{b}_{\text{tot}}^\dagger$ M times and normalizing:

$$|\psi(0)\rangle = \frac{(\hat{b}_{\text{tot}}^\dagger)^M}{\sqrt{M!}} |\text{vacuum}\rangle \tag{3.5}$$

Fermionic search The fermionic search has Hamiltonian,

$$\hat{H}_{\text{fermion}} = -\gamma_c \sum_{i,j\in V} A_{ij}\hat{f}_i^\dagger \hat{f}_j - \hat{f}_w^\dagger \hat{f}_w \tag{3.6}$$

We cannot simply apply M uniform creation operators like $\hat{b}_{\text{tot}}^\dagger$ on the vacuum state, as this might violate the Pauli exclusion principle. To get our fermionic initial state, we use the Slater determinant. To construct a uniform state, we choose single-particle states that have equal amplitude on each vertex. Furthermore, they need to be normalized and orthogonal. The following set of states suffices:

$$|\phi_k\rangle = \frac{1}{\sqrt{N}} \sum_{j\in V} e^{\frac{jk}{N}2\pi i} |j\rangle \text{ for } k = 1, \ldots, M. \tag{3.7}$$

We construct our initial state as described in 2.3.4, using the Slater determinant.

3.3. The critical hopping rate for an arbitrary graph

We assume that the critical hopping rate γ_c of our multi-particle quantum search, the hopping rate for which the running time of the search is minimized, is equal to the critical hopping rate of the single-particle quantum search on that graph. We acknowledge that this hypothesis may be incorrect. Simulations of the quantum search algorithm using different hopping rates provided in section 5.3 will support or disprove this hypothesis.

Thus, the problem becomes that of finding the critical hopping rate γ_c of the single-particle quantum search,

$$\hat{H}_{\text{search}} = -\gamma A - |w\rangle\langle w|, \tag{3.8}$$

where A is the adjacency matrix of the graph.

In [1] the optimal hopping rate γ_c' for a different quantum search model, namely

$$\hat{H}_{\text{search}}' = \gamma H + |w\rangle\langle w|, \tag{3.9}$$

was found to be

$$\gamma_c' = S_1 = \sum_{i=1}^{N-1} \frac{|\langle w|v_i\rangle|^2}{1-\lambda_i}, \tag{3.10}$$

where H is a matrix encoding the structure of the graph with eigenvalues $0 \le \lambda_1 \le \cdots \le \lambda_{N-1} < \lambda_N = 1$ and v_i is its eigenvector corresponding to eigenvalue λ_i. One such matrix H is the adjacency matrix of the graph, which is real and symmetric, so has real eigenvalues. Normalizing this matrix, we can ensure all eigenvalues are between -1 and 1. By adding a diagonal matrix and dividing by 2, we obtain a matrix with eigenvalues between 0 and 1. This optimality for γ given by (3.10) holds only when the *spectral condition* is met, which is treated in 3.3.2.

3.3.1. Conversion of γ_c between the two search hamiltonians

Now that we know the critical hopping rate γ_c' of $\hat{H}_{\text{search}}'$, we want to know the critical hopping rate γ_c for $\hat{H}_{\text{search}}$. To this end, we assume the spectral condition is met and start with the optimal quantum search

$$\hat{H}_{\text{search}} = S_1 H + |w\rangle\langle w|, \tag{3.11}$$

We choose as our graph-encoding matrix, the adjacency matrix A. To make sure all eigenvalues are between 0 and 1, we normalize A as follows:

$$H = \frac{1}{2}\Big(\frac{A}{\|A\|^2} + I\Big), \tag{3.12}$$

where $\|A\| = \max(|\lambda_i(A)|)$. Substituting gives,

$$\hat{H}_{\text{search}} = \frac{S_1}{2\|A\|^2} A + |w\rangle\langle w| + \frac{S_1}{2} I \tag{3.13}$$

We will show that the I-term in this Hamiltonian does not change the dynamics of the search. Consider two search Hamiltonians H and $H' = H + cI$, $c \in \mathbb{R}$. Starting in the same initial state, the evolution under H' becomes

$$|\psi'(t)\rangle = e^{-iH't}|\psi(0)\rangle = e^{-icIt}e^{-iHt}|\psi(0)\rangle = e^{-ict}|\psi(t)\rangle. \tag{3.14}$$

Thus H' introduces a global phase and does not change the probabilities as compared with evolution under H: H' and H result in the same dynamics, and must therefore have the same optimal hopping rate. This means that,

$$\hat{H}_{\text{search}} = \frac{S_1}{2\|A\|^2} A + |w\rangle\langle w| \tag{3.15}$$

is optimal as well.

To make the final step, we consider search Hamiltonians H and $H' = -H$. Starting again at the same initial state, we get

$$|\psi'(t)\rangle = e^{-iH't}|\psi(0)\rangle = e^{+iHt}|\psi(0)\rangle = |\psi(-t)\rangle\,. \tag{3.16}$$

Therefore, evolution under H' is the same as evolution under H in reverse time. One can show that reversing the time does not influence the dynamics of the quantum search. To prove this, we consider the projection of the state $|\psi'(t)\rangle$ (the state evolved under H' for time t) onto a state $|\phi\rangle$,

$$\langle\phi|\psi'(t)\rangle = \langle\phi|\,e^{+iHt}|\psi'(0)\rangle = \Big(\langle\psi'(0)|\,e^{-iHt}|\phi\rangle\Big)^* \tag{3.17}$$

If we again assume the same initial state, so $|\psi(0)\rangle = |\psi'(0)\rangle$, and consider the probability of measuring $|\phi\rangle$ after a time t for the state $|\psi'(t)\rangle$, we get

$$|\,\langle\phi|\psi'(t)\rangle\,|^2 = |\Big(\langle\psi'(0)|\,e^{-iHt}|\phi\rangle\Big)^*|^2 = |\,\langle\psi(0)|\,e^{-iHt}|\phi\rangle\,|^2 = |\,\langle\psi(t)|\phi\rangle\,|^2. \tag{3.18}$$

Thus, we see that the probability of measuring $|\phi\rangle$ is the same for both states. If the probability of measuring any state is the same for both $|\psi'(t)\rangle$ and $|\psi(t)\rangle$, we can conclude that if H' is optimal for some γ, then H is optimal for that same γ. Therefore,

$$\hat{H}_{\text{search}} = -\frac{S_1}{2\,\|A\|^2}A - |w\rangle\langle w| \tag{3.19}$$

is optimal as well. Comparing with the search Hamiltonian of our thesis, we see that the optimal hopping rate γ_c for our model is given by,

$$\gamma_c = \frac{S_1}{2\,\|A\|^2}. \tag{3.20}$$

3.3.2. The spectral condition

The spectral condition states that

$$\sqrt{\epsilon} < c\min\left\{\frac{S_1S_2}{S_3}, \Delta S_2\right\} \tag{3.21}$$

has to be satisfied for some small constant $c > 0$*. Here $\sqrt{\epsilon} = \langle w|v_N\rangle$, with v_N the eigenvector of H from (3.9) corresponding to its largest eigenvalue λ_N, $\Delta = \lambda_N - \lambda_{N-1}$, and S_1, S_2 and S_3 given by

$$S_k = \sum_{i=1}^{N-1}\frac{|\,\langle w|v_i\rangle\,|^2}{(1-\lambda_i)^k}, \quad k = 1,2,3. \tag{3.22}$$

According to Theorem 1 of [1], if this condition is satisfied and $\gamma = S_1$, and furthermore $S_1/\sqrt{S_2} = \Theta(1)$†, then the quantum search of (3.9) is optimal. With optimal, we mean that the state in the quantum search reaches a nonzero amplitude at the marked vertex in as short a time as possible. In [3] it is shown that a quantum search can reach nonzero amplitude in at least $T = \Omega(1/\sqrt{\epsilon})$ time. In Theorem 1 of [1] it is shown that, if the conditions just mentioned are met, the state does in fact reach a nonzero amplitude at the marked vertex in time $T = \Theta(1/\sqrt{\epsilon})$, and therefore it is optimal.

*__Interpretation of c.__ The constant $c > 0$ has to be chosen small enough for an argument in [1] to work (smaller than 0.2, as will be mentioned later). We could not find a physical interpretation for this c. However, given a certain graph, we can calculate the value of c for which both sides of (3.21) are equal, written as $\tilde{c}$. Say for a graph $\tilde{c} > c$ (so for example $\tilde{c} = 0.4$), then the spectral condition is not satisfied. This $\tilde{c}$ is calculated for multiple types of graphs and plotted in figure 3.1. We see that for sufficiently large graphs ($N > 10$), $\tilde{c}$ tends to be smaller if the graph is more connected. Thus, $\tilde{c}$ seems to depend on the connectivity of a graph.

†In this thesis we use the following asymptotic notation: $T = \Omega(n)$ means T grows at least as fast as n. $T = o(n)$ means T grows less fast than n. $T = \Theta(n)$ means T grows neither slower nor faster than n.

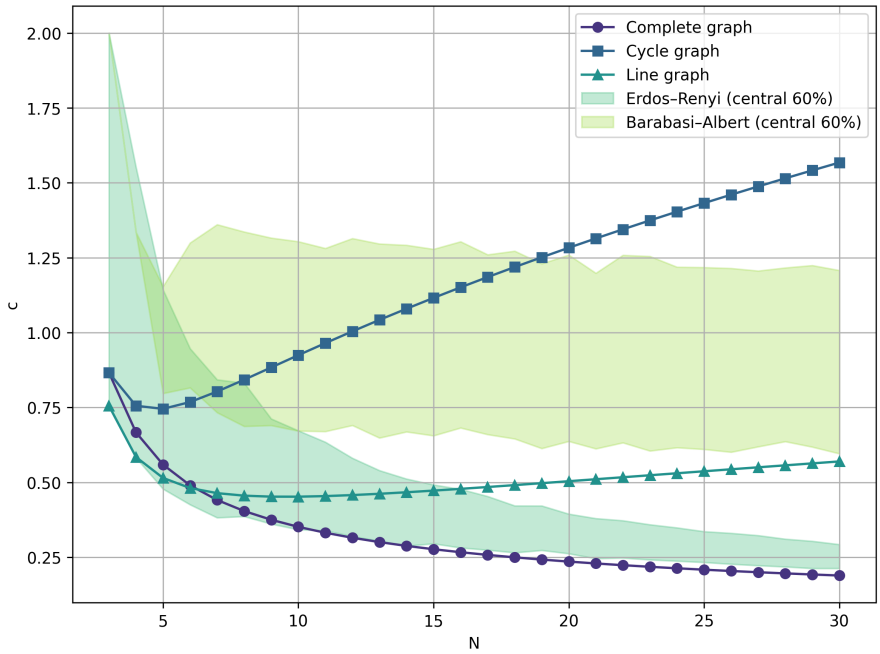


Figure 3.1: Plot of the value of c for which the left and right-hand side of the spectral condition are equal, as a function of N for different graph types. For the Erdős–Rényi and the Barabási–Albert graph type, for each N, 1000 instances were analyzed, of which the central 60% are indicated with the colored region.

Whether the spectral condition is met depends on the structure of the graph and on the location of the marked vertex in the graph. Furthermore, it depends on how small the constant c has to be. It can be shown that, for the argument that is used to prove Theorem 1 in [1] to work, c needs to be smaller than 0.2. This is explained briefly in [9]. To determine if the spectral condition is met for the graphs that will be simulated, we calculate the value of c for which the left and the right-hand side of the spectral condition are equal. These values are plotted in 3.1 for different graph types as a function of N. For all c larger than this value, the spectral condition is met. The graphs that will be used in this thesis contain at most 30 vertices. As can be seen in 3.1, the vast majority of these only satisfy the spectral condition if $c > 0.2$, which is not allowed. Thus, we can conclude that the spectral condition is not satisfied for the graphs used in this thesis.

For the graphs used in this thesis, we cannot say whether the quantum search of (3.9) is optimal, in the sense that it might not reach nonzero amplitude at the marked vertex in $T = \Theta(1/\sqrt{\epsilon})$ time. However, we assume that in [1] the authors choose γ (the γ in (3.9)) so that the search is as fast as possible, even when the spectral condition is not satisfied. Although the search might not be optimal for this γ when the spectral condition is not satisfied, it may still be as fast as possible. With this, we mean that the quantum search reaches a nonzero amplitude at the marked vertex in as short a time as possible, though maybe not within a time $T = \Theta(1/\sqrt{\epsilon})$. This assumption may not hold in general, and will therefore be tested as well by simulating the quantum search for multiple γ. In the main simulations, however, the hopping rate given by (3.20) will be used.

4

Overhead marked vertex identification

Simply evolving the quantum system just defined is not enough to find the marked vertex. After evolution, we need to measure the system, and based on this measurement, produce a guess (or multiple guesses) for the marked vertex. Alternatively, we could evolve the system multiple times and base our guess on multiple measurements. Therefore, the total quantum search algorithm consists of a quantum evolution part, followed by a classical marked vertex identification part. In the following chapter, we will describe our method of overhead marked vertex identification (meaning identification of the marked vertex after measuring the quantum system).

4.1. The total runtime of a quantum search algorithm

Translating our model into code and simulating the quantum search algorithm on a graph, we see that the output of our algorithm is a quantum superposition of states. Each state is a Fock state, a certain distribution of particles over the vertices. Now, in a real-world scenario, this superposition would be unknown. In order to get information out of the system, it has to be measured. Upon measurement, the superposition collapses into one of the states.

4.1.1. Classical oracle calls

To what state the superposition collapses is determined by the amplitude of each state in the superposition. If our quantum search algorithm works well, it will draw the particles towards the marked vertex during the evolution of the system. This means that the amplitudes of the Fock states that have more particles in or near the marked vertex will grow. Then, at a time of choice, we stop the evolution. Very rarely does the system evolve into a state with all particles on the marked vertex. In fact, for the fermionic algorithm, this is impossible because of the Pauli exclusion principle, preventing more than one fermion from occupying the same vertex. Therefore, upon measurement of the system, we might end up with a distribution where another vertex has more particles on it than the marked vertex! If this is the case, the marked vertex might instead be the vertex with the second-largest occupation, or third. To find the marked vertex, we would have to perform oracle calls on the vertices from high occupation to low occupation, that is, manual classical calls.

4.1.2. Multiple evolutions

Another trick one could use is to independently evolve the system again from the same initial state and measure it. If, in a fermionic search, a certain vertex ends up occupied, this does not say much, as there will be M vertices occupied by a fermion. If, however, one evolves the system again and that same vertex ends up occupied, then this will more likely be the marked vertex. In this way, adding a measurement amplifies the probability that our guess for the marked vertex will be correct. We will call this probability amplification. How much this probability is amplified, depends on the quantum state of the system upon measurement.

The total runtime T of the search algorithm is then determined by how often one evolves and measures the system ($n_{\text{evolutions}}$), how long each evolution takes ($t_{\text{evolution}}$) and how many oracle calls one per-

forms (n_{oracles}). In this thesis, for each independent evolution of the system, we measure it at the same time $t_{\text{evolution}}$. The time the total search algorithm takes to produce a guess for the marked vertex, which we will call the runtime T, is,

$$T = n_{\text{evolutions}} \cdot t_{\text{evolution}} + n_{\text{oracles}} \cdot t_{\text{oracle}} \tag{4.1}$$

where t_{oracle} is the time it takes to perform one oracle call, which we consider a predetermined constant. This expression for the total runtime is similar to the expression used in [1]. In this thesis however, for simplicity, we decide to use only one oracle call. The last term in this expression for T then becomes $n_{\text{oracles}} \cdot t_{\text{oracle}} = t_{\text{oracle}}$. As t_{oracle} is assumed to be the same for the bosonic search as for the fermionic search, we can ignore it, as we will only compare these two algorithms. The expression for T thus becomes

$$T = n_{\text{evolutions}} \cdot t_{\text{evolution}} \tag{4.2}$$

What remains is the seemingly complicated task of finding $n_{\text{evolutions}}$ and $t_{\text{evolution}}$ such that T is optimal.

What we want to accomplish ultimately, after all those evolutions and oracle calls, is finding the marked vertex. This cannot be guaranteed, as measuring the system produces a random state. However, the chance P_{success} that the total search algorithm (so including multiple evolutions and oracle calls) will eventually lead to the marked vertex, can often be made arbitrarily high. Thus $P_{\text{success}} >$ [some probability threshold of choice] is the constraint that we have to satisfy when trying to minimize T.

4.2. The bosonic pile-up advantage

4.2.1. The hypothesis

As mentioned earlier, the fermionic search has a disadvantage. Due to the Pauli exclusion principle, at most one fermion can occupy any vertex. This means that after a measurement of the system, there will always be ambiguity as to which vertex should be the marked vertex, as M vertices will be occupied. The bosonic search does not have this disadvantage. Any number of bosons can occupy a vertex. As the Hamiltonian draws the bosons to the marked vertex, they will pile up there, making the marked vertex easy to distinguish. This is called the **bosonic pile-up effect**, and it leads us to believe that the bosonic search should work faster than the fermionic search. This hypothesis was worked out in [6] for bosonic and fermionic searches on a complete graph in the large graph ($N \to \infty$) and large particle ($M \to \infty$) limit, using the same Hamiltonian as in this thesis. Moreover, the same initial state was taken as in this thesis, and the hopping rate was chosen $\gamma = \frac{1}{N}$, which is the critical hopping rate for a complete graph as predicted by [1] and (3.20) in the large N limit. Importantly, the method for marked vertex identification in [6] is different from the one used in this thesis, sometimes using multiple oracle calls or a different method of probability amplification. For the rest of this section, we will consider the search algorithm of [6], unless otherwise stated.

4.2.2. Quantum search on a complete graph in the large particle limit

In [6] they found that, both for bosonic and fermionic search, the time it takes for such a system to evolve into a state where the marked vertex has a nonzero probability to be occupied by a particle scales with $\sqrt{N/M}$. Immediately after this evolution, one would check each vertex containing a particle using an oracle call, so that the number of oracle calls scales with M. (In [6] this is called the naive method: the method without using probability amplification.). Using (4.1) and assuming the number of evolutions to be constant, we get

$$T = \Theta(\sqrt{N/M}) + \Theta(M) = \Theta(\sqrt{N/M} + M) \tag{4.3}$$

.

4.2.3. Analysis of the bosonic runtime scaling

However, the assumption that one has to check every vertex containing a particle may be incorrect because of the bosonic pile-up effect. In fact, this hypothesis is proven in [6] using the following argument.

A key result of the analysis of the bosonic search in [6] is that whenever $M \ll \sqrt{N}$, the number of bosons on the marked vertex follows a Poisson distribution, with mean $\lambda = \chi^2$,

$$P_k(\chi) = e^{-\lambda}\frac{\lambda^k}{k!}, \quad \lambda = \chi^2. \tag{4.4}$$

Here, the number of bosons on the marked vertex is k and χ is a measure of the time of evolution $t_{\text{evolution}}$. To be precise, $\chi = t_{\text{evolution}}\sqrt{M/N}$. The number of bosons on any other vertex follows a Poisson distribution as well, but with mean $\mu = \frac{M}{N-1} \ll 1$. We see that as time progresses and χ increases, the Poisson distribution of the marked vertex shifts to the right, which means that more particles occupy it on average. Furthermore, for any other vertex, the Poisson distribution has a very small mean μ, meaning that few particles will occupy it.

Using this result, we determine the probability that two or more bosons occupy the marked vertex.

$$\begin{aligned} P[n_w \geq 2] &= 1 - P[n_w = 0 \text{ or } n_w = 1] \\ &= 1 - (e^{-\lambda} + e^{-\lambda}\lambda) \\ &= 1 - \left(e^{-\chi^2} + e^{-\chi^2}\chi^2\right) \end{aligned} \tag{4.5}$$

Here n_w stands for the number of bosons on the marked vertex. This chance becomes greater than $1/2$, whenever $\chi \simeq 1.296$. Now, the chance that there exists another vertex with two or more bosons is

$$P\big[\text{there exists a vertex } i \ (i \neq w) \text{ s.t. } n_i \geq 2\big] \tag{4.6}$$

$$\begin{aligned} &= 1 - P\big[\text{for all vertices } i \ (i \neq w)\text{: } n_i < 2\big] = 1 - (e^{-\mu} + e^{-\mu}\mu)^{N-1} \\ &= \frac{M^2}{2N} + O\left(\frac{M^3}{N^2}\right) \end{aligned} \tag{4.7}$$

Again, n_i is the number of particles on vertex i. When $M \ll \sqrt{N}$, this probability is $< 1/2$.

Given these two facts: that (1) the probability that the marked vertex contains two or more bosons is $> 1/2$, and (2) that the probability that no other vertex contains two or more bosons is also $> 1/2$, we define the rule: "Declare the unique vertex with $n_i \geq 2$ to be the marked vertex"*. Now we evolve and measure the system multiple times, and for each measurement note which vertex gets assigned the marked vertex via this rule. Consequently, we take the vertex that gets assigned to be the marked vertex the most frequent as our guess. According to [6], the probability that our guess is correct can be amplified to any desired probability in this way. Thus, after evolving the system multiple times and declaring the marked vertex using this method, only one oracle call is needed. The $\Theta(M)$-term in (4.3) vanishes, and the expression for the bosonic search becomes

$$T = \Theta(\sqrt{N/M}). \tag{4.8}$$

If one optimizes the runtime over the number of bosons by taking $M = \Theta(\sqrt{N})$, this becomes

$$T = \Theta(N^{1/4}). \tag{4.9}$$

Note that it is assumed here that the number of evolutions needed, $n_{\text{evolutions}}$, does not scale with N or M, or at least does not dominate the scaling of the runtime.

4.2.4. Analysis of the fermionic runtime scaling

This argument cannot be made for the fermionic search, as $P[n_i \geq 2] = 0$ for any vertex i of course. Therefore, in [6], it is stated that in order to find the marked vertex from a measurement of the system, one has to classically check each vertex containing a fermion. Thus the $\Theta(M)$-term in (4.3) cannot be ignored. If again the

*Upon measurement, there may be multiple vertices containing two or more bosons. In this case, all vertices with two or more bosons have to be checked by an oracle call. In [6] it is assumed that the time it takes to perform these extra oracle calls, does not dominate the runtime scaling.

runtime scaling is minimized over the number of particles by taking $M = \Theta(N^{1/3})$, the following expression is obtained:

$$T = \Theta(N^{1/3}). \tag{4.10}$$

One can conclude that the runtime of the bosonic search scales less fast with N compared to the fermionic search, which means it works better on complete graphs in the large N limit.

4.3. Our method of overhead marked vertex identification

4.3.1. Motivation for further analysis

The method for overhead marked vertex identification of [6] exploits the bosonic pile-up advantage of the bosonic search. As can be expected, such a bosonic search algorithm outperforms a fermionic search without any probability amplification. If we would like to compare bosonic and fermionic search, we should for each search try to find a method that exploits its particular advantages or characteristics. The task of finding the best method of overhead marked vertex identification for each search (bosonic and fermionic) may be a thesis in itself, so we will propose a method that will at least not completely inhibit the fermionic search, whilst still making use of the bosonic pile-up advantage.

It is true that after one measurement of a fermionic search, there are still M candidates for the marked vertex. If, however, we evolve the system three times, and three times we measure that vertex i contains a particle, while no other vertex is occupied multiple times, then this makes vertex i likely to be the marked vertex. This argument is especially convincing when the number of vertices is large compared to the number of fermions. We will call our method of overhead marked vertex identification the majority vote method, and it exploits this idea. From now on we consider arbitrary graphs again, and leave the setting of the large M, large N limit.

4.3.2. The majority vote method

If one evolves the system $n_{\text{evolutions}}$ times and measures each system after a time $t_{\text{evolution}}$, one ends up with $n_{\text{evolutions}}$ distributions of M particles over N vertices. From these measurements we want to obtain a guess for the marked vertex. Since the Hamiltonian draws particles towards the marked vertex, it is natural to look at the sum of the particles of the different measurements on each vertex, and take a majority vote over these sums. The vertex with the largest total number of particles is our guess for the marked vertex. More concretely, we can write a **single measurement** of the system as $(n_1, n_2, \ldots, n_N) = \mathbf{n}$, where n_i is the number of particles measured on vertex i. The **total measurement** after multiple evolutions is then $(n_1^{(1)}, \ldots, n_N^{(1)}), (n_1^{(2)}, \ldots, n_N^{(2)}), \ldots, (n_1^{(r)}, \ldots, n_N^{(r)}) = \mathbf{n^{(1)}}, \ldots, \mathbf{n^{(r)}}$, where we have abbreviated $n_{\text{evolutions}}$ as r: the number of rounds in the majority vote. Summing the measurements, we get $(n_1^{(1)} + \cdots + n_1^{(r)}, \ldots, n_N^{(1)} + \cdots + n_N^{(r)}) = \mathbf{n^{(1)}} + \cdots + \mathbf{n^{(r)}}$, so that our guess for the marked vertex is

$$\tilde{w} \text{ is vertex } i \text{ such that } n_i^{(1)} + \cdots + n_i^{(r)} > n_j^{(1)} + \cdots + n_j^{(r)} \text{ for all } j \neq i \tag{4.11}$$

If there are multiple vertices that have the largest total amount of particles, then the majority vote method produces no guess for w. In this case, we are always wrong. We could have also let our algorithm check all these vertices that have the largest total amount of particles using oracle calls. Although this might improve our algorithm, we choose not to add this extra step to the algorithm, for simplicity.

Implementing the majority vote method on r rounds of measurements is easy. We just calculate the sums of the occupations of each vertex and find the maximum. However, if we want to determine how well the majority vote method works for a quantum search on a certain graph, this is more complex. Specifically, we would like to know for a general quantum state $|\psi\rangle$ and a given marked vertex w what the probability is of guessing the right marked vertex using r rounds of the majority vote method. In other words, if $|\psi\rangle$ is our quantum state and w our marked vertex, what is the chance that after r measurements of $|\psi\rangle$, the vertex i with $n_i^{(1)} + \cdots + n_i^{(r)} > n_j^{(1)} + \cdots + n_j^{(r)}$ for all $j \neq i$, is indeed vertex w. We will call this probability the success probability $P_r(\psi)$. Since the state $|\psi(t)\rangle$ is a function of time, we will sometimes write $P_r(t)$.

4.4. Determining the success probability of the majority vote method

We present two ways of determining the success probability of the majority vote method, the first exact and the second approximate. The exact method calculates the success probability as the expectation value of a quantum operator. If $|\psi\rangle$ is our state, we can project it unto all Fock basis states where the marked vertex has the largest occupation, to get the projected state $|\phi\rangle$. Then the success probability of a majority vote of one round will be $|\langle\psi|\phi\rangle|^2$. This idea is explained in detail in appendix B. Whilst this method enables fast calculation of $P_r(\psi)$, it takes up a lot of memory. In our code, a general fermionic state on a graph of N vertices is stored using 2^N complex numbers. A majority vote operator of r rounds works on the total state, which is this fermionic state tensored r times with itself. This total state therefore requires 2^{Nr} complex numbers to be stored. Applying a majority vote of five rounds on a fermionic search over ten vertices would require 9 PB of memory, which is beyond modern processor capabilities.

Instead of calculating the success probability exactly, we can estimate it. Say we have a quantum state $|\psi\rangle$ and we want to estimate the success probability for this state after a majority vote of r rounds. We can do this by measuring the state r times (in the computer simulation that is), and repeating this n times. We end up with n samples of r measurements, with each sample looking as follows

$$\text{one sample: } \begin{cases} \mathbf{n^{(1)}} & = (n_1^{(1)}, \dots, n_N^{(1)}) \\ \vdots & \\ \mathbf{n^{(r)}} & = (n_1^{(r)}, \dots, n_N^{(r)}) \end{cases}.$$

Here $n_i^{(j)}$ is the number of particles that was measured on vertex i in the j-th measurement. Then, on each sample we perform a majority vote to find out what our guess for the marked vertex would be, based on that sample. We keep track of how frequently our guess is correct and call this number s_n, the number of successes.

There is an underlying probability that a sample will produce a correct guess, namely the success probability $P_r(\psi)$. The k-th sample can therefore be seen as a realization of a Bernoulli random variable X_k with success probability $p = P_r(\psi)$. In order to estimate this p, we use the maximum likelihood estimator for a Bernoulli random variable, which is $\hat{p} = S_n/n$, where $S_n = X_1 + X_2 + \cdots + X_n$. As we take more samples, this estimate becomes more accurate. How accurate exactly can be determined via Hoeffding's inequality [5], using that $\mathbb{E}(\hat{p}) = p$,

$$P\Big(\frac{|S_n - \mathbb{E}(S_n)|}{n} \geq t\Big) \leq 2e^{-2nt^2} \tag{4.12}$$

$$P\Big(\Big|\frac{S_n}{n} - \mathbb{E}\Big[\frac{S_n}{n}\Big]\Big| \geq t\Big) \leq 2e^{-2nt^2} \tag{4.13}$$

$$P\big(|\hat{p} - p| \geq t\big) \leq 2e^{-2nt^2}. \tag{4.14}$$

Therefore, the exact value of $P_r(\psi)$ will lie within a distance t from our estimate s_n/n with probability $1 - 2e^{-2nt^2}$. In this thesis we want to know with 99.99% certainty that our estimate is at most 0.01 away from the true value of $P_r(\psi)$, which means we have to take $n = 5.0 \cdot 10^4$ samples.

This estimation method takes 10-100 times longer to compute as compared to the exact method, but uses significantly less memory. In practice the size of the graph and the number of particles one can simulate is now determined by the size of the quantum state, and not dependent on the number of rounds in the majority vote. A fermionic state is stored as 2^N complex numbers and a bosonic state as $(M+1)^N$ complex numbers. Thus a typical processing core in DelftBlue with 4GB of memory could theoretically run a fermionic search over 20 vertices or a bosonic search of 7 bosons over 7 vertices, assuming 400 time-points (so 400 quantum states).

If we are only interested in at what time the success probability crosses a certain threshold, say 0.8, then we do not have to know $P_r(\psi)$ at every time-point with 0.01 accuracy. We only want to know if we can say with 99.99% certainty whether $P_r(\psi)$ is above, below or within 0.01 distance of the threshold. This means we can

take less samples for some time-points, which, when implemented in our simulation, results in a 4x speed-up.

Which method we use to determine the success probability, exact or approximate, depends on the type of computer that is available. In the case of a supercomputer there is enough computing power. The limiting factor is processor memory. Therefore, we will use the approximate method.

5

Analyzing the performance of a quantum search algorithm

Now that we have defined our total search algorithm, we will describe how the performance of such an algorithm can be measured. This measure enables us to analyze the performance of bosonic and fermionic quantum search algorithms.

5.1. A measure for the performance of a quantum search algorithm

5.1.1. First results of the simulation

The quantum search as defined in chapter 3 together with the method for overhead marked vertex identification of chapter 4 were implemented in a parallel computer program and run on the DelftBlue supercomputer. Using this program, the success probability over time, $P_r(t)$, of a quantum search algorithm can be estimated. To make the distinction, we will write $\hat{P}_r(t)$ for our estimate and $P_r(t)$ for the actual success probability. To give an idea of how $\hat{P}_r(t)$ can look, we have run two simulations. The first is a bosonic search algorithm of $M = 2, 3,$ and 4 particles over a complete graph of $N = 5$ vertices (this graph is shown in figure 5.1). The second is a fermionic search algorithm of $M = 2, 3$ and 4 particles over the same graph. For both search algorithms, $\hat{P}_r(t)$ was calculated with a majority vote of $r = 1, 2, \ldots, 12$ rounds and for 400 time points between $t = 0$ and $t = 40$. The estimated bosonic success probabilities are shown in figure 5.2, the estimated fermionic success probabilities are shown in figure 5.3. Dashed vertical lines indicate the times for which $\hat{P}_r(t)$ rises above the probability threshold of 0.8 for the first time [*].

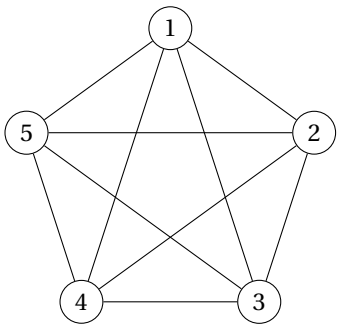


Figure 5.1: A complete graph of $N = 5$ vertices, where vertex 1 is the marked vertex. The quantum searches of figures 5.2 and 5.3 were performed over this graph.

[*] For the fermionic search of $M = 3$ particles over $N = 5$ vertices, shown in figure 5.3, the dashed line indicating this time for $\hat{P}_6(t)$ can be seen at $t \approx 15$ in between peaks of the oscillation. Clearly, at $t \approx 15$, $\hat{P}_6(t)$ is well below the threshold. This dashed line is here because $\hat{P}_6(t)$ just touches the threshold in the peak to the left of $t \approx 15$ and finally crosses the threshold in the peak to the right, resulting in an average of $t \approx 15$.

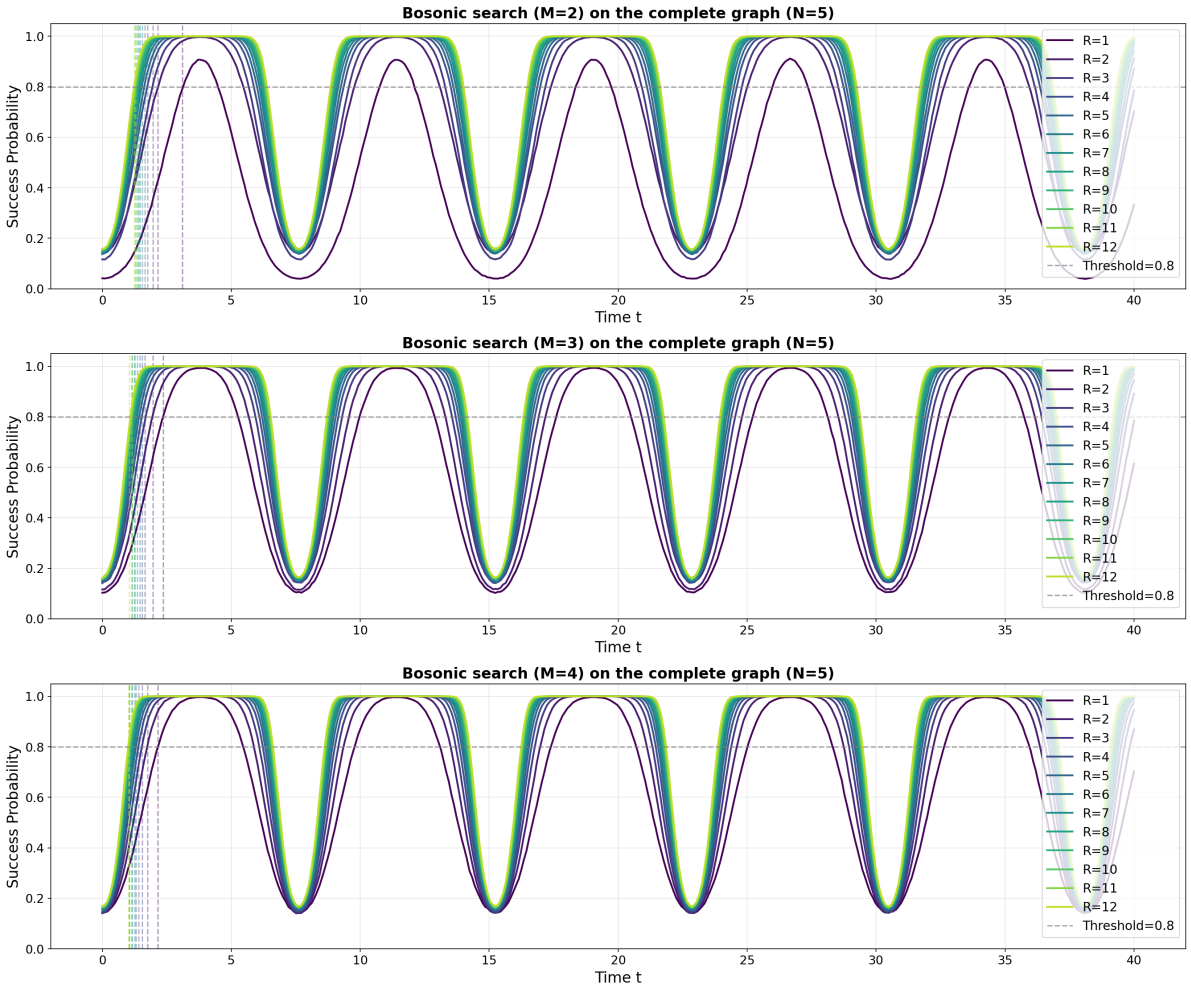


Figure 5.2: The estimated success probability as a function of time for a bosonic search with $M = 2,3$, and 4 bosons (displayed in the top, middle and bottom plot respectively) over a complete graph of $N = 5$ vertices. In each plot $\hat{P}_r(t)$ is displayed for $r = 1,2,\ldots,12$ and $t \in [0,40]$. A horizontal, dashed, gray line indicates the probability threshold of 0.8. The vertical, dashed, colored lines indicate the times at which $\hat{P}_r(t)$ rises above this threshold for the first time.

We see that the estimated success probability undergoes oscillatory motion. As the number of rounds of the majority vote increases, the majority vote method tends to amplify the estimated success probability. In this way, $\hat{P}_1(t), \hat{P}_2(t), \ldots$ and $\hat{P}_{12}(t)$ oscillate together, that is, with the same phase. This amplification diminishes as the number of rounds increases. Whereas a two-round majority vote in the bosonic search of $M = 2$ bosons amplifies the success probability substantially ($\hat{P}_2(t)$ is substantially larger on average than $\hat{P}_1(t)$, see figure 5.2), for larger numbers of rounds the success probability seems to converge. Whether $P_r(t)$ converges as $r \to \infty$, and if so, to what function, is an interesting problem that is not treated in this thesis. For some search algorithms, the majority vote method reduces the success probability (see the fermionic search of $M = 2$ fermions in figure 5.3, looking at the lowest two curves between $t = 7$ and $t = 9$). Note that for all fermionic searches (of at least 2 fermions), $P_1(t) = 0$ for all $t > 0$. This is because upon measurement, there are always at least two vertices occupied, so that the majority vote will result in a tie, which leads to no guess for the marked vertex at all. It can be shown that in general, $P_r(t)$ is not periodic, though a derivation is beyond the scope of this thesis.

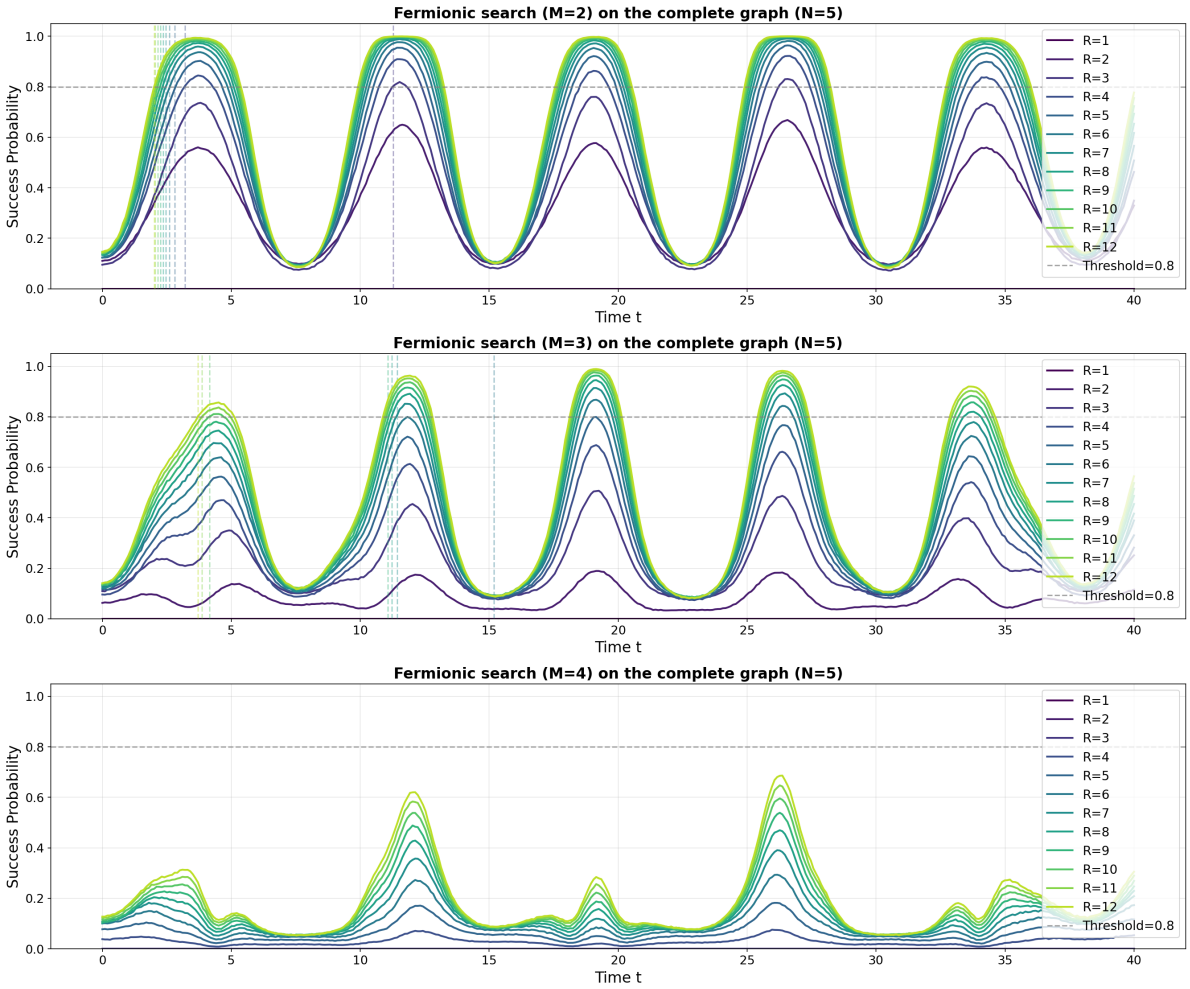


Figure 5.3: The estimated success probability as a function of time for a fermionic search with $M = 2, 3,$ and 4 fermions (displayed in the top, middle and bottom plot respectively) over a complete graph of $N = 5$ vertices. In each plot $\hat{P}_r(t)$ is displayed for $r = 1, 2, \ldots, 12$ and $t \in [0, 40]$. A horizontal, dashed, gray line indicates the probability threshold of 0.8. The vertical, dashed, colored lines indicate the times at which $\hat{P}_r(t)$ rises above this threshold for the first time.

5.1.2. The runtime of our quantum search algorithm and its error

Ultimately, if we are interested in the performance of a search algorithm, we want to know how long that algorithm takes to find its target. For a quantum search algorithm, this time is undefined, as one can never know for sure whether the algorithm will find its target (the marked vertex) within a certain time. Therefore, we define the runtime of a quantum search algorithm to be the time it takes the algorithm to find its target with some minimal probability. In this thesis, we set this probability to be 0.8. This runtime, then, is our measure for the performance of a quantum search algorithm. Consider, for example, the fermionic search of $M = 2$ fermions over the complete graph of $N = 5$ particles in figure 5.3 using a $r = 3$ rounds majority vote. The curve of $\hat{P}_3(t)$ (the second lowest visible curve in the plot[†]) crosses the probability threshold at $t \approx 11$. We will call this time the **crossing time** t^*: the time at which $\hat{P}_r(t)$ rises above the threshold for the first time. This is the time at which we would measure the system, and repeat the evolution and measurement r times. Therefore, the runtime T of this algorithm becomes $T = r \cdot t^* \approx 3 \cdot 11 = 33$, using (4.2).

From the simulation data we can extract an interval in which the actual runtime T has to lie with some probability. As mentioned earlier, we estimate the success probability $P_r(t)$ until there is 0.9999 probability that the actual value lies within 0.01 from our estimate. If we consider the time period where $\hat{P}_r(t)$ crosses the threshold (so near t^*) and zoom in on the probabilities near the threshold, we see a situation similar to the one sketched in figure 5.4. The error bars indicate the probability interval where we know $P_r(t)$ lies with 0.9999 certainty, the center of each error bar being our estimate $\hat{P}_r(t)$. Looking at this sketch, we see that if indeed $P_r(t)$ lies in the indicated interval for all $t \in \{10.2, 10.6, 11.0, 11.4, 11.8\}$, then $P_r(t)$ has to cross the threshold between $t = 10.6$ and $t = 11.8$, so that the crossing time t^* has to lie in the interval $(10.6, 11.8)$ (assuming this is the first time $\hat{P}_r(t)$ rises above the threshold). This is the method used to determine the interval for the crossing time in our simulation. If $P_r(t)$ lies within 0.01 distance of our estimate $\hat{P}_r(t)$ for all simulated t, then we know that t^* lies in this interval. Since we simulate for 400 time-steps, the probability that $P_r(t)$ lies within 0.01 distance from our estimate for all t is $(0.9999)^{400} \approx 0.96$, so we know with at least 0.96 certainty that the crossing time lies within its interval. This is a conservative lower bound for the probability that t^* lies within its interval. For low $\hat{P}_r(t)$'s, say $\hat{P}_r(t) < 0.6$, even though $P_r(t)$ may be more than 0.01 away from its estimate $\hat{P}_r(t)$, $P_r(t)$ is still very unlikely to be near the probability threshold of 0.8, so that it will never influence the crossing time. Thus, the probability that t^* lies within the interval calculated by the program will be larger than 0.96.

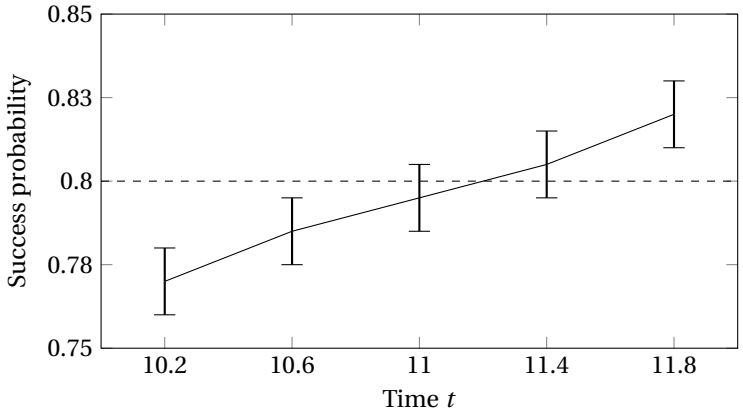


Figure 5.4: A sketch of the success probability over time, for times near t^* and probabilities near the threshold of 0.8 (indicated by the horizontal dashed line). The center of each error bar is the estimated $\hat{P}_r(t)$ and the interval indicates the region in which $P_r(t)$ lies with 0.9999 certainty.

5.1.3. The problem with determining the runtime

There are two problems with our measure for quantum search performance. To illustrate these, consider a plot of the success probabilities for a fermionic search with $M = 4$ fermions over a Erdős–Rényi graph of $N = 6$ generated with $p = 0.6$, see figure 5.5. Note that $\hat{P}_r(t)$ was plotted for $t \in [0, 40]$ and $r \in \{1, 2, \ldots, 12\}$, because it

[†]As $\hat{P}_1(t) = 0$, because this is a fermionic search.

is computationally impossible to simulate the state for $t \in [0,\infty)$ and estimate the success probabilities for all $r \in \{1,2,3,\ldots\}$. Furthermore, suppose that we had estimated only the success probabilities for $r \in \{1,2,\ldots,11\}$ (so excluding 12) and $t \in [0,20]$. Then, according to our simulation, the success probability of the search algorithm would never cross the threshold, so that we could say nothing about its performance. Say we ran the simulation for $t \in [0,40]$ instead, but still only estimating for $r \in \{1,2,\ldots,11\}$. Then $P_r(t)$ would cross the threshold at $t^* \approx 25$ (using a majority vote of $r = 9$ rounds), so that the runtime would be $T \approx 9 \cdot 25 = 225$. However, if we did estimate for $r = 12$ as well, the success probability would cross the threshold already at $t^* = 5$, so that the runtime would be $T \approx 12 \cdot 5 = 60$! We see that the runtime according to the simulation depends on the time for which we simulate and the numbers of rounds for which we estimate. One can, therefore, never be sure of the runtime of our quantum search algorithm, unless one simulates for all $t > 0$ and $r \in \mathbb{N}_{\geq 1}$.

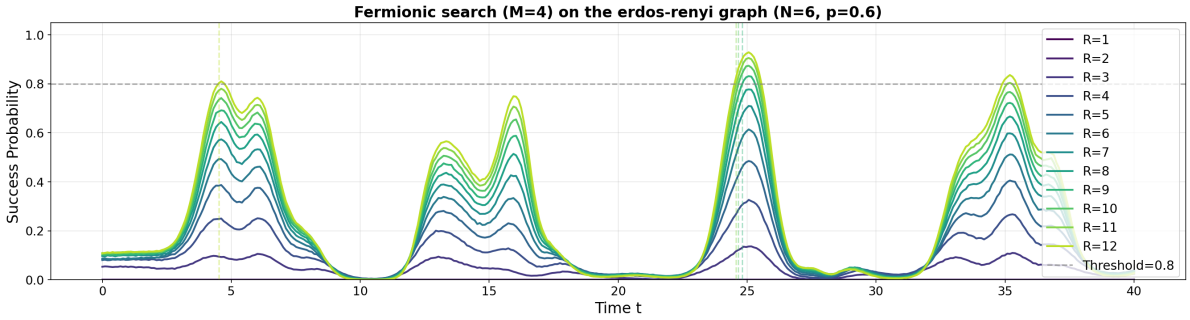


Figure 5.5: A plot of $\hat{P}_r(t)$ for a fermionic search with $M = 4$ fermions over an Erdős–Rényi graph of $N = 6$ vertices, generated with $p = 0.6$, for multiple $r = 1,2,\ldots,12$.

5.1.4. Validity of the determined runtime

Let us investigate in what manner the runtime, calculated by the simulation, changes as we estimate $P_r(t)$ for larger r. In figure 5.6 the estimated success probability of a bosonic search ($N = 5$, $M = 2$) and a fermionic search ($N = 4$, $M = 2$) over a complete graph is plotted for the same t and r as before. However, now the calculated crossing time for each r is indicated with a vertical line that is colored as follows: as we estimate $P_r(t)$ for larger r starting with $r = 1$, we only color the line indicating its crossing time if the corresponding runtime (i.e. $r \cdot t^*$) is smaller than all runtimes calculated so far (so for all $r' = 1,2,\ldots,r-1$). All other crossing times are indicated with a dotted, vertical, gray line. In this way, we get an idea of how the runtime changes as we estimate $P_r(t)$ for larger r.

Note that in figure 5.6 for the bosonic search, the threshold is already reached for $r = 1$ in the first peak of the oscillation, as indicated by the colored vertical line at $t \approx 3$. For the following $r = 2,3,\ldots$, the time t^* that $\hat{P}_r(t)$ rises above the threshold for the first time, decreases slightly for every increment of r, but not enough to compensate for the factor r in $T = r \cdot t^*$ (which increases with 1 every time), so that $T = r \cdot t^*$ does not decrease. Consequently, these crossing times are indicated with dotted, gray lines. In the fermionic search in figure 5.6, $\hat{P}_4(t)$ crosses the threshold in the fifth peak of the oscillation. Increasing r by one, the estimated success probability $\hat{P}_5(t)$ crosses the threshold two peaks earlier. This decrease in t^* is sufficiently large to compensate for the r-term in $T = r \cdot t^*$ which increases with one, so that T decreases. Hence this crossing time is indicated by a colored vertical line. Increasing r once more, the time at which $\hat{P}_6(t)$ crosses the threshold shifts to the first peak, decreasing T again. For all larger r (up to $r = 12$), the runtime does not decrease anymore.

We hypothesize that this behavior holds for all of our quantum search algorithms, that is, that the runtime T can only decrease as we increase r, if the crossing time t^* shifts one or multiple peaks to the left. If the crossing time decreases slightly, but stays on the same peak as we increase r (like in the bosonic search of figure 5.6), we hypothesize that this decrease is too small to compensate for the r-term in $T = r \cdot t^*$, which increases with one, so that T does not decrease. If this hypothesis were true, we would only need to estimate $P_r(t)$ for larger and larger r until the first peak of the oscillation (as seen in the plot of $\hat{P}_r(t)$) rises above the threshold. Then, the time at which $\hat{P}_r(t)$ crosses the threshold for the first time, can never shift one or multiple peaks to the left, as there are no more peaks to the left.

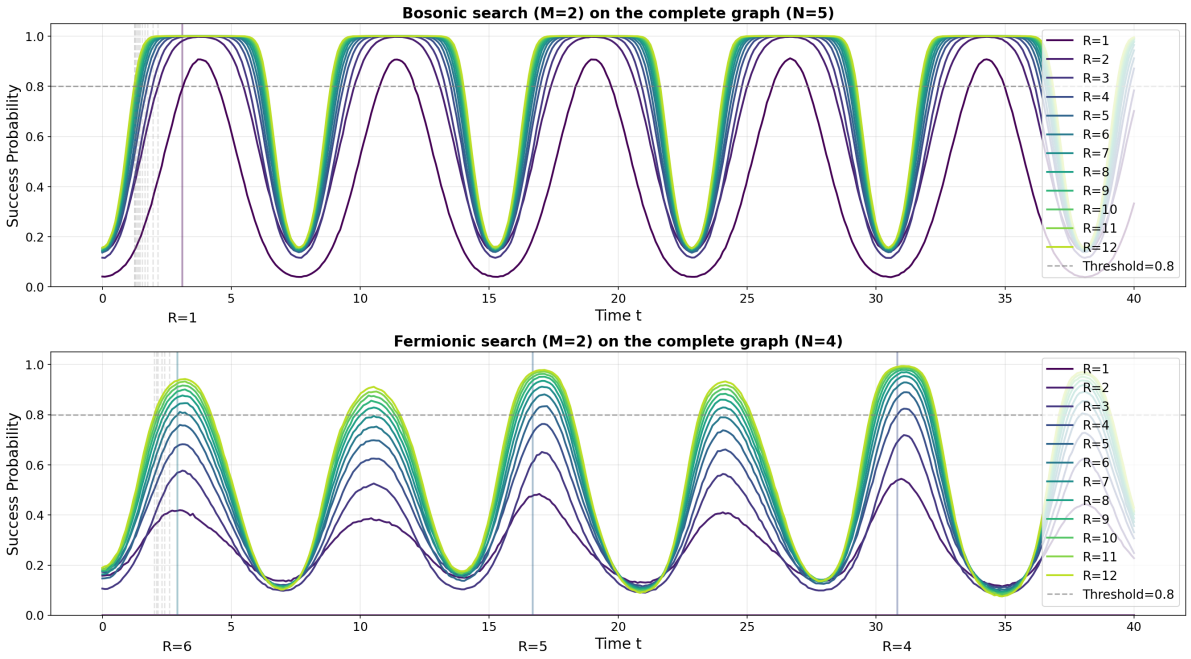


Figure 5.6: A plot of $\hat{P}_r(t)$ for a bosonic search with $M = 2$ bosons over a complete graph of $N = 5$ vertices (upper plot) and a fermionic search with $M = 2$ fermions over a complete graph of $N = 4$ vertices (lower plot). The probability threshold of 0.8 is again indicated with a horizontal, dashed line. The crossing time of each r is now only indicated with a vertical, colored line if the corresponding runtime T is lower than all previously calculated runtimes (so for $r' = 1, 2, \ldots, r-1$). Otherwise the crossing time is indicated with a vertical, dashed, gray line.

This hypothesis was checked manually for 136 fermionic quantum search algorithms (all search algorithms analyzed in 5.2), for $r = 1, 2, \ldots, 12$. For these search algorithms, it can be seen in the plots of $\hat{P}_r(t)$ that indeed the runtime T only decreases when the crossing time t^* shifts at least one peak to the left. This was verified for r up to $r = 12$, and our assumption is that this behavior continues for all $r > 12$. Furthermore, for these quantum searches it was verified that for $r = 12$ the first peak of $\hat{P}_r(t)$ rises above the threshold. This means that, for these 136 search algorithms, we reason that the runtime as calculated by the simulation is correct, as the crossing time cannot shift one or more peaks to the left. In other words, we assume that one may estimate $P_r(t)$ for rounds $r = 13, 14, 15, \ldots, 100$ or more, but this will only decrease t^* slightly (as t^* stays within the same oscillation), so that T will never decrease. We implicitly assume here, that the first oscillation of $\hat{P}_r(t)$ is visible in the plot, and that there will not appear another earlier peak as we estimate $P_r(t)$ for larger r.

5.2. Scaling of the fermionic runtime

Estimating the success probabilities and validating the calculated runtimes manually as described in section 5.1.4, we can obtain an interval that contains the runtime with at least 96% confidence, making the assumptions stated in section 5.1.4. This interval was calculated for all fermionic search algorithms over the complete graph of $N = 3, 4, \ldots, 16$ vertices. For each of these N, a fermionic search was simulated of $M = 2, 3, \ldots, N-1$ fermions (for $M = 1$, bosonic and fermionic search is the same, and for $M = N$, all vertices are always occupied, so that $P_r(t) = 0$). For each N the fermionic search algorithm of M fermions and r rounds was selected with the lowest calculated runtime[‡]. These lowest runtime intervals were plotted as a function of N in figure 5.7. One can see that the lowest runtime for $N = 4$ is $T \approx 17$, and is achieved in a fermionic search with $M = 2$ fermions and using a majority vote of $r = 6$ rounds.

Notice first that each lowest runtime (for $N = 3, 4, \ldots, 16$) is achieved using a fermionic search with $M = 2$ particles. Secondly, we see that the plot shown in figure 5.7 does not follow a clear trend. At first sight, the lowest runtime seems to decrease as a function of N, which is the opposite of what one would expect of a search

[‡]Here the runtime is taken to be the average of the minimum and maximum of its runtime interval

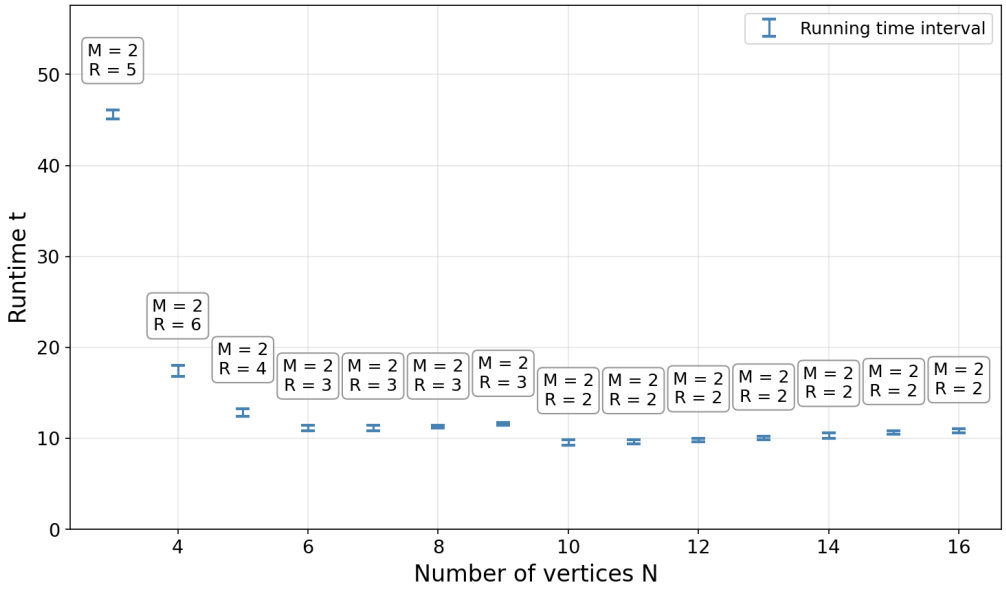


Figure 5.7: A plot of the runtime interval of a fermionic search over a complete graph of N vertices, as a function of N. For each N, the fermionic search using M fermions and a r rounds majority vote was taken, so that this runtime interval was lowest. This M and r is shown above each corresponding runtime interval.

algorithm over a graph, when this graph gets larger (when N increases). However, when we look closer, we see that the N-domain can be split up into several sections. For $N = 3,4,5$, the lowest runtime decreases fast and for every N the search algorithm that is fastest uses a different number of rounds, $r = 5,6,4$ respectively. For $N = 6,7,8,9$, the search algorithm that is fastest uses $r = 3$ rounds, and the lowest runtime increases slightly. Then from $N = 9$ to $N = 10$, the lowest runtime drops again, and we enter the final section. From $N = 10$ up to $N = 16$, the fastest search algorithm uses $r = 2$ rounds and increases again as a function of N.

We hypothesize that this increasing trend of the runtime as a function of N within each section, and the decrease of the runtime over the whole simulated N-domain, can be seen as the result of two opposing effects. One effect influences the average height of $\hat{P}_r(t)$ and the other influences the frequency of $\hat{P}_r(t)$.

5.2.1. Influence of M/N on the average height of $\hat{P}_r(t)$

From its plot of $\hat{P}_r(t)$, the performance of our quantum search algorithm can be estimated visually by the average height of $\hat{P}_r(t)$. If $\hat{P}_r(t)$ is high on average, it will need less rounds r to cross the threshold, so that $T = r \cdot t^*$ will be smaller and the performance of this search algorithm better. Consider for example the fermionic searches of $M = 2$ and $M = 3$ on the complete graph of $N = 5$ vertices in figure 5.3. The search using $M = 2$ particles looks higher on average than the search of $M = 3$ particles. Indeed, the search of $M = 2$ particles only needs $r = 3$ rounds to cross the threshold of 0.8, whilst the search of $M = 3$ particles needs $r = 6$ rounds. Consequently, the runtime of the fermionic search using $M = 2$ particles is lower than the one using $M = 3$ particles.

The figure 5.3 suggests that as we use more fermions, the function $\hat{P}_r(t)$ is less high on average, so that we need more rounds r to cross the threshold and the performance of the search algorithm worsens. To investigate how the runtime of a fermionic search changes as we use more fermions, a plot was made of the runtime of a fermionic search over a complete graph of $N = 16$ vertices, as a function of M, see figure 5.8. For each M, r was chosen such that this runtime was lowest and this r is shown above each runtime interval. In this plot we see that indeed the number of rounds necessary to cross the threshold increases with M. As a consequence, the runtime T increases as well, confirming our theory[§]. It was verified, using the simulations computed by DelftBlue, that for $N = 1,2,\ldots,16$ the runtime increases as one uses more fermions. This ex-

[§]The runtime interval corresponding to $M = 11$ is much wider than the other intervals. This is because in this fermionic search, the curve $\hat{P}_{12}(t)$ just touches the threshold at $t \approx 6$ and finally crosses the threshold multiple peaks later at $t \approx 32$, leading to a wider runtime interval.

plains why, in figure 5.7, each lowest runtime is achieved by a fermionic search with $M = 2$ fermions, as this is the lowest number of fermions possible (for $M = 1$, bosons and fermions act the same).

We hypothesize that, more generally, for a fermionic search $\hat{P}_r(t)$ is higher on average as M/N decreases. This can be supported briefly by the following idea. From figure 5.2 and figure 5.3 we see that $\hat{P}_r(t)$ is higher on average in bosonic searches as compared to fermionic searches on the complete graph[¶]. Treating the particles classically, when M/N is small, the particles encounter each other less frequently, since the graph is large and the number of particles small. In our quantum search, this might mean that the particles interact less with each other. Since bosons and fermions only differ by how they interact with each other, this would mean that as M/N gets smaller, bosons and fermions behave more and more the same. For a fermionic search, this would mean that as M/N gets smaller, the search would look more like a bosonic search, so that its estimated success probability would be higher on average.

For our best fermionic search (the one using $M = 2$ particles), M/N decreases as the size N of the complete graph gets larger. If our hypothesis were true, this would mean that $\hat{P}_r(t)$ will be higher on average as N increases. Consequently, as N increases, the best fermionic search would need less rounds, which is exactly what we see in the plot of figure 5.7. As N gets larger, $\hat{P}_r(t)$ becomes higher on average, until for some N the number of rounds necessary to cross the threshold decrements and the lowest runtime $T = r \cdot t^*$ decreases sharply. This can be seen in figure 5.7 in between sections. Whenever the number of rounds decreases (for example from $N = 9$ to $N = 10$), the runtime drops.

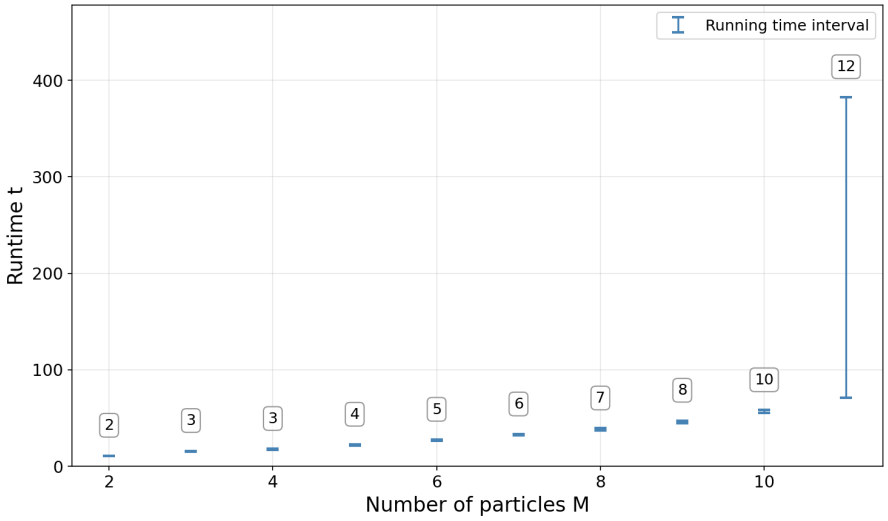


Figure 5.8: A plot of the runtime intervals for a fermionic search over a complete graph of $N = 16$ vertices, as a function of M. M for which the fermionic search has a runtime interval that is unbounded are not included. Above each runtime interval the number of rounds of the majority vote for which the search algorithm was fastest, is shown.

5.2.2. Influence of N on the frequency of $\hat{P}_r(t)$

Increasing N therefore can decrease the runtime $T = r \cdot t^*$ by decreasing r via the effect described in 5.2.1. The opposing effect causes t^* to increase as a function of N, so that the runtime T increases, and it has to do with the frequency of $\hat{P}_r(t)$. To explain this second effect, let us consider $\hat{P}_r(t)$ for a fermionic search with $M = 2$ fermions over complete graphs of $N = 4, 10$ and 16 vertices, see figure 5.9. Counting the number of oscillations in each plot, we see that as N increases, the number of oscillations decreases, that is, the frequency of $\hat{P}_r(t)$ decreases. In [6] a similar effect is discussed for fermionic search over a complete graph in the large-N limit. It is proven that for this search, the probability of finding the marked vertex to be occupied by a fermion (so after one measurement) has frequency

$$\omega = \sqrt{\Omega^2 + \Delta^2} = \sqrt{\frac{M}{N} + \frac{M^2}{N^2}}. \tag{5.1}$$

[¶]From figure 5.2 and figure 5.3 this follows for $N = 5$. However, this was verified for all $N = 3, 4, 5$.

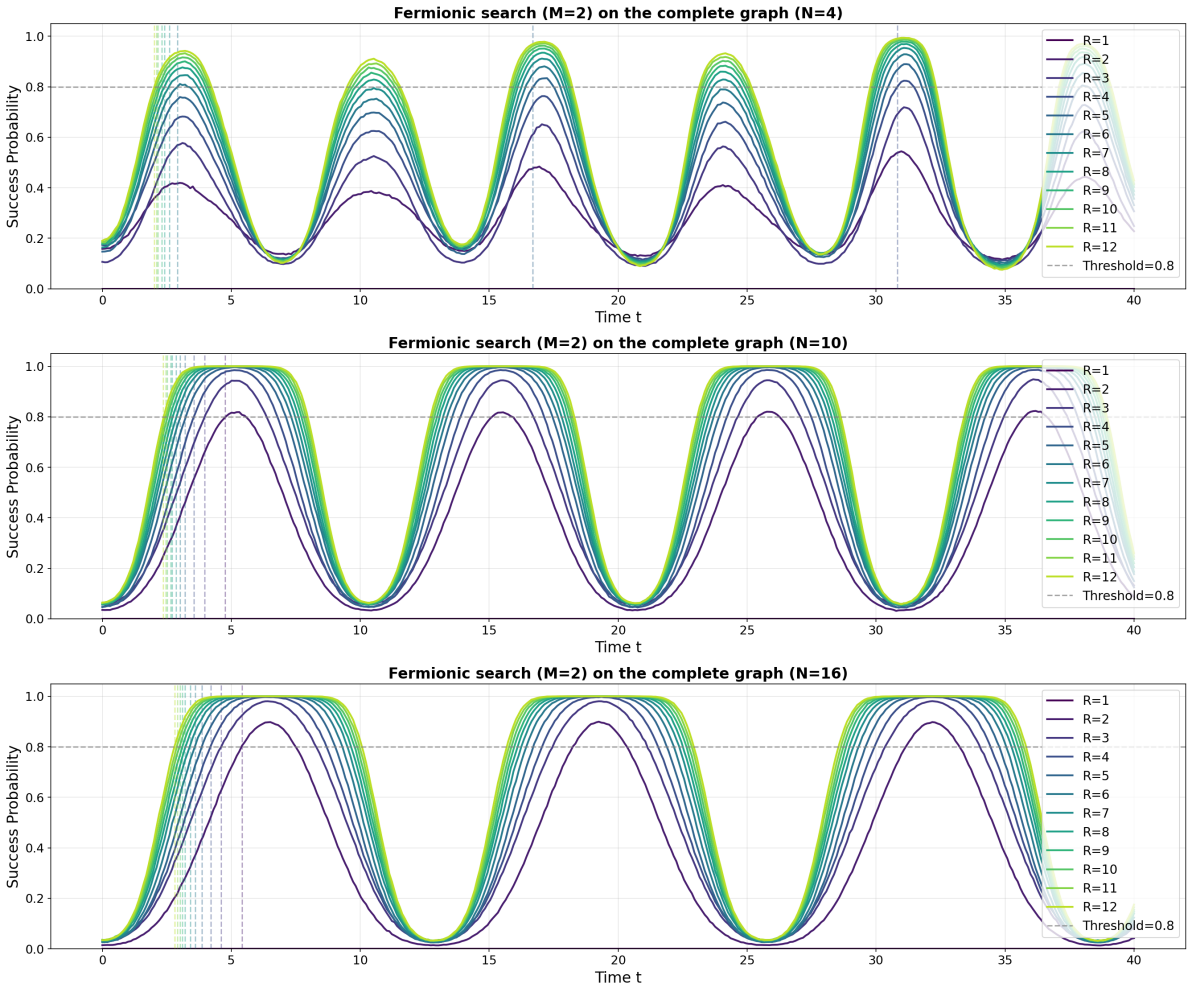


Figure 5.9: A plot of $\hat{P}_r(t)$ for a fermionic search using $M = 2$ fermions over a complete graph of $N = 4, 10$ and 16 vertices (displayed in the upper, middle and lower plot respectively). The crossing time for each r is indicated with a vertical, colored, dashed line. The probability threshold of 0.8 is indicated with a horizontal, dashed, gray line.

This probability is not equal to $P_r(t)$, as it does not include the majority vote method or any other method for overhead marked vertex identification. However, we hypothesize that since $P_r(t)$ is clearly related to this probability, its frequency will show similar scaling with N. Furthermore, this probability was calculated in the large-N regime, and not for the regime of the data presented in figure 5.7, namely $N < 17$. Nevertheless, making these assumptions, we conclude that the frequency of $\hat{P}_r(t)$ must decrease as N increases.

What is the effect of a decreasing frequency of $\hat{P}_r(t)$ on the runtime? When the frequency of $\hat{P}_r(t)$ decreases, one oscillation takes more time to complete, so that $\hat{P}_r(t)$ rises and falls more gradually and each peak in the oscillation shifts slightly to the right. If each peak shifts to the right, then the time t^* at which $\hat{P}_r(t)$ rises above the threshold shifts to the right as well, causing $T = r \cdot t^*$ to increase. This increase of t^* is gradual, as opposed to the large discrete decrements of r, so that its effect on the runtime as a function of N is a gradual increase. Indeed, whenever r is constant (so in each section), the runtime in figure 5.7 increases gradually.

5.2.3. Analyzing the scaling of the fermionic runtime

To analyze the runtime of this fermionic search as a function of N, we could fit it to some function. In [6], it is shown that without a method for overhead marked vertex identification, this runtime should scale as $\Theta(N^{1/3})$. The same bosonic search as used in this thesis, but with a different method for overhead marked vertex identification, was shown to have a runtime that scales as $\Theta(N^{1/4})$. In this thesis we decide to fit the fermionic runtimes to the function $T(N) = \alpha N^{\beta}$ with parameters α and β determined via least-squares fit-

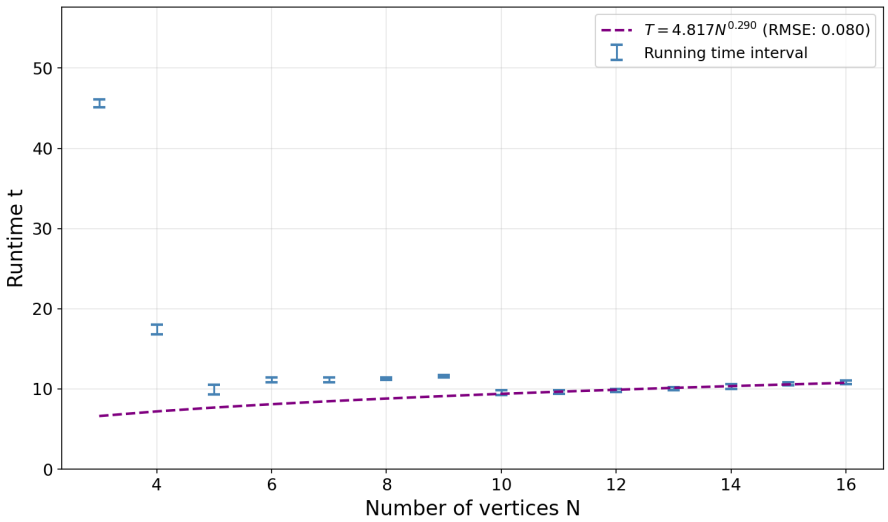


Figure 5.10: A plot of the lowest runtime interval of a fermionic search over a complete graph of N vertices, as a function of N. The dashed, purple curve indicates the function $T(N) = \alpha N^{\beta}$ fitted to the runtime intervals of $N = 10, 11, \ldots, 16$, using the method of least-squares, giving $\alpha = 4.817$ and $\beta = 0.290$ with a root mean square error of 0.080.

ting[‖].

For $N > 16$, a decreasing M/N should no longer influence the runtime. Even though the average height of $\hat{P}_r(t)$ may still increase, this will no longer lead to a decrease in r. This is because one needs at least a two-round majority vote, since $P_1(t) = 0$ for any fermionic search (with $M \geq 2$), which is already achieved from $N = 10$ onwards. Thus, unless some other effect is overlooked, the only remaining effect for $N > 16$ will be that of a decreasing frequency of the estimated success probability. Therefore, we expect that for $N > 16$, the runtime will continue to increase gradually, following the trend of $N = 10, 11, \ldots, 16$, without sudden drops like the one from $N = 9$ to $N = 10$.

If we are interested in the scaling of the runtime for large N, we should therefore use only the data from $N = 10, 11, \ldots, 16$ to perform a least-squares fit on, and ignore the runtimes for $N < 10$. Doing this, we obtain the fitted function $T(N) = 4.817N^{0.290}$ which has root mean square error 0.080. Thus, according to this fit the runtime of a fermionic search over a complete graph scales as $\Theta(N^{0.290})$ which is still faster than for the bosonic search, but slower than was predicted in [6] for the fermionic search. However, in order to produce a sound estimate for the parameter β, fermionic simulations for larger N are needed.

5.3. Optimality of the critical hopping rate

In all simulations so far, the hopping rate was set to the critical hopping rate for a single-particle quantum search as predicted by [1]. In order to assess whether this hopping rate was indeed optimal, multiple bosonic searches of $M = 2$ bosons over a complete graph of $N = 5$ vertices were simulated. Instead of using the critical single-particle hopping rate of [1], the search evolved using multiple hopping rates in the range $\gamma \in [0.10, 0.45]$. Plotting the resulting runtime intervals leads to the plot shown in figure 5.11. The same was done for a fermionic search with $M = 2$ fermions over a complete graph of $N = 5$ vertices, but only for $\gamma \in [0.10, 0.35]$, and shown in figure 5.12.

In both plots, the critical single-particle hopping rate of [1] is indicated with a vertical red line at $\gamma \approx 0.17$. The hopping rate used in [6] is indicated with a vertical green line at $\gamma = 0.20$. The hopping rate that was found to be the optimal hopping rate for each search in this simulation, that is, the hopping rate for which the runtime interval was lower than that for all other $\gamma \in [0.10, 0.45]$ or all $\gamma \in [0.10, 0.35]$, is indicated with a vertical purple line. We see that for the bosonic search, the optimal hopping rate was found to be $\gamma = 0.24$ which is larger than the critical hopping rate predicted by [1]. This could mean that the interactions between

[‖]As runtime data, the center of each runtime interval was taken, and the function was fitted with respect to these points.

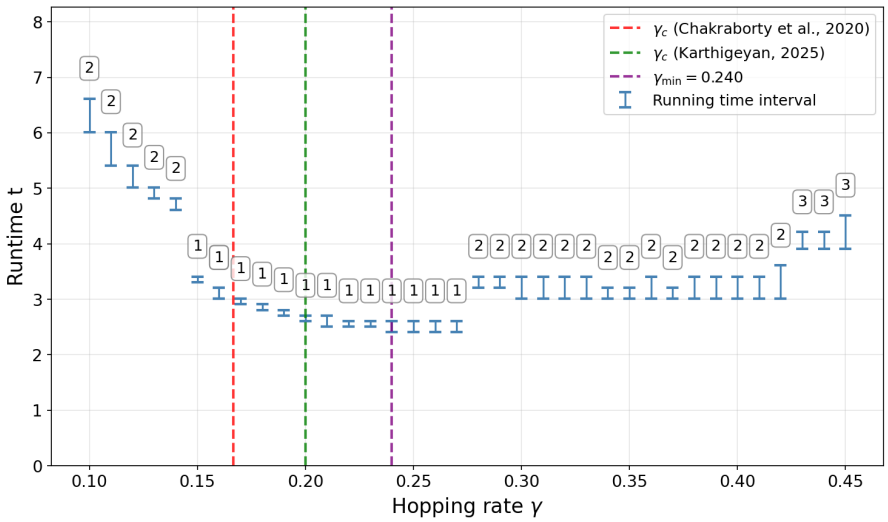


Figure 5.11: A plot of the runtime intervals for a bosonic search with $M = 2$ bosons over a complete graph of $N = 5$ vertices as a function of the hopping rate γ. Above each runtime interval, the number of rounds is displayed for which the runtime interval is lowest for that particular hopping rate. The red, vertical line indicates the single-particle critical hopping rate as predicted by [1]. The green, vertical line indicates the hopping rate used in [6]. The purple, vertical line indicates the hopping rate $\gamma = 0.240$ for which the bosonic search was found to be fastest (at least among the hopping rates that were simulated).

particles in a bosonic (or fermionic) search change the hopping rate for which the search is optimal. This conclusion is further supported by the fact that the plot of the runtime as a function of the hopping rate looks different for the bosonic search as compared to the fermionic search, even though both searches take place using the same number of particles and over the same graph. Furthermore, it could mean that if the small constant c in the argument of [1] is larger than 0.2 (which is the case for the complete graph of $N = 5$ vertices), then because the spectral condition is not satisfied, the critical single-particle hopping rate is not given by $\gamma_c = S_1$. Note that for the fermionic search, the hopping rate used in [6] is in fact equal to the hopping rate that was found optimal as compared to the other $\gamma \in [0.10, 0.35]$ for which the search was simulated. However, this might be a coincidence.

It is clear that the hopping rate that was used in our model for a multi-particle quantum search, was not the hopping rate for which this search is optimal. This makes comparison between bosonic and fermionic searches, and even between two fermionic searches, less reliable. However, there is currently no alternative for the critical single-particle hopping rate of [1]. What the critical hopping rate for a multi-boson or multi-fermion quantum search is, is as yet an interesting open problem.

5.4. Influence of graph connectivity on the runtime

To investigate the influence of graph connectivity on the runtime of our quantum search algorithm, 256 searches of $M = 2$ particles were simulated on Erdős–Rényi graphs of $N = 6$ vertices, half of which were bosonic searches, the other half fermionic. Each Erdős–Rényi graph was generated using a p drawn from a uniform distribution between 0.4 and 1.0. Only graphs that are connected were used, and for each graph the average shortest distance between two vertices was calculated. For the bosonic searches and for the fermionic search two scatterplots were created with the runtime on the y-axis and the average shortest path length on the x-axis, see 5.13.

As can be seen in 5.13, as the average shortest path length of a graph increases, the variation in the runtime of a search increases as well. Furthermore, we see that the average runtime appears to increase slightly with the average shortest path length as well, though too little graphs with a large average shortest path length (say > 1.6) were simulated to say for sure. Comparing bosonic with fermionic search, we see that the bosonic searches are on average significantly faster than the fermionic searches. Moreover, looking at the average shortest path lengths smaller than 1.4, we observe that the variation in the runtime seems to grow faster for fermionic searches than for bosonic searches (the points in the scatterplot spread out more quickly in the right plot). This would mean that for Erdős–Rényi graphs (and possibly graphs in general) of $N = 6$ vertices,

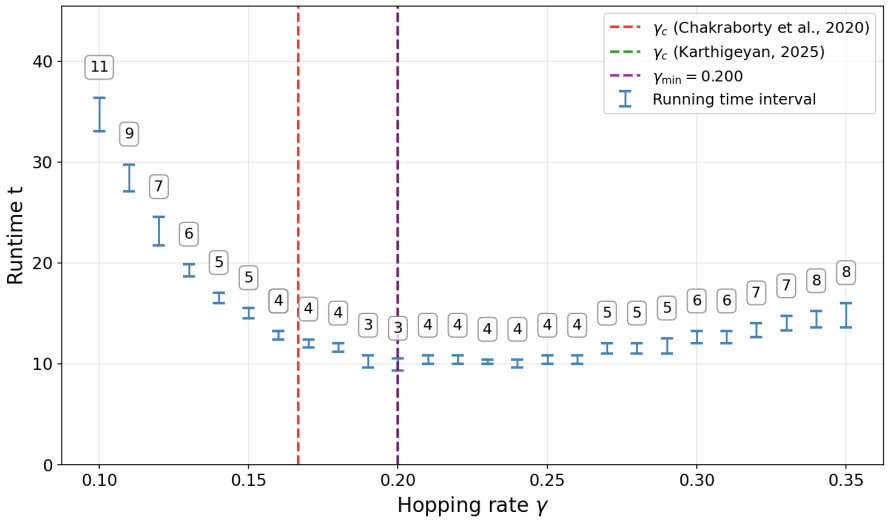


Figure 5.12: A plot of the runtime intervals for a fermionic search with $M = 2$ fermions over a complete graph of $N = 5$ vertices as a function of the hopping rate γ. Above each runtime interval, the number of rounds is displayed for which the runtime interval is lowest for that particular hopping rate. The red, vertical line indicates the single-particle critical hopping rate as predicted by [1]. The green, vertical line indicates the hopping rate used in [6]. The purple, vertical line indicates the hopping rate $\gamma = 0.200$ for which the fermionic search was found to be fastest (at least among the hopping rates that were simulated).

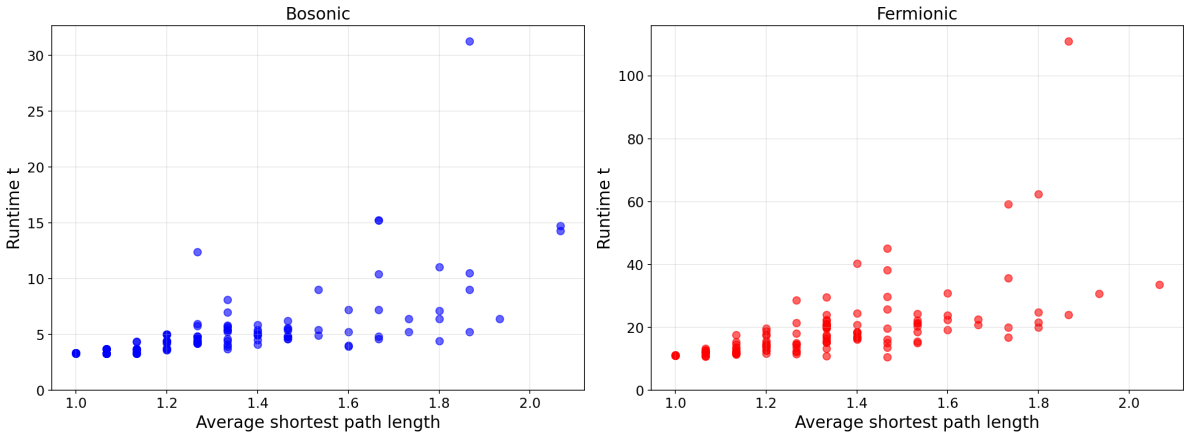


Figure 5.13: Two scatterplots of the runtime of a search against the average shortest path length of the graph over which the search took place. Left is the plot for all bosonic searches, and right is the plot for the fermionic searches.

the performance of a fermionic search is less consistent than the performance of a bosonic search. However, further data is required to draw definitive conclusions.

6

Conclusion

In this thesis, we have studied continuous-time quantum search algorithms using bosons and fermions on arbitrary graphs. We used the many-particle search Hamiltonian from [6] with the single-particle hopping rate from [1], and developed a method for overhead marked vertex identification. The performance of the resulting search algorithm was analyzed via simulations run on the Delftblue supercomputer. Finally, the validity of the simulation results was discussed.

The runtime of a fermionic search over a complete graph of N vertices as a function of N was analyzed. It was found that this runtime increases gradually with N but drops sharply at some N. It was argued that as N increases, the frequency of the success probability of the search decreases, so that the runtime increases gradually. Furthermore, it was reasoned that as M/N decreases, the average height of the success probability increases, so that for some N less rounds in the majority vote are needed, causing the runtime to drop sharply. Finally, the runtime was fitted to $T(N) = \alpha N^{\beta}$ and β was found to be 0.290. This would mean that the runtime of the fermionic search scales less fast than predicted by [6].

The runtime of bosonic and fermionic search over a complete graph as a function of its hopping rate γ was analyzed. It was found that the critical single-particle hopping rate from [1] is not necessarily optimal for multi-particle quantum search. Furthermore, it was shown that the optimal hopping rate for bosonic search and fermionic search can differ, even though the same number of particles are used and the search is performed over the same graph.

Finally, the runtime of bosonic and fermionic search over Erdős–Rényi graphs as a function of the average shortest path length of the graph was analyzed. We saw that, as the average shortest path length of a graph increases, the variation in the runtime of a quantum search increases as well. For fermionic searches this variation appeared to be greater, implying that the performance of a fermionic search is less consistent than the performance of a bosonic search on graphs that are less connected.

We can conclude that, for the quantum search algorithms analyzed in this thesis, bosonic search is substantially faster and possibly more robust than fermionic search. However, the results from the simulation suggest that, using our method for overhead marked vertex identification, the runtime scaling of the fermionic search may improve substantially as compared to [6].

Discussion We provide some comments regarding the validity and possible errors of this thesis. Furthermore, we suggest interesting directions for further research.

In section 5.1.4, the validity of the runtime, as calculated in the simulation, is discussed. In short, we hypothesize that whenever the first peak of the oscillation of $\hat{P}_r(t)$ crosses the threshold, the lowest runtime is reached. This means that, for all higher r that are not yet estimated in the program, the runtime would be higher than the lowest runtime found so far. Although all 136 simulations inspected so far follow our hypothesis for r up to $r = 12$, it is not guaranteed that this behavior will continue for r up to infinity. Thus, the runtime as calculated in the simulation could be wrong.

In section 5.2.3 the fermionic runtime ($M = 2$) over a complete graph was analyzed as a function of N and fitted to $T(N) = \alpha N^{\beta}$, where β was determined to be 0.290. This fit was based on seven runtimes calculated by the simulation. Although these data follow the fit well, more runtimes are needed to see if the trend of the runtime that is set for $N = 10, 11, \ldots, 16$ continues, and to determine this β more precisely.

In section 5.3 it was shown that the single-particle hopping rate that is used in our quantum search is not optimal. Since bosonic search and fermionic search may have a different optimal hopping rate, using the single-particle hopping rate may impede one of these searches more than the other. Therefore, until this optimal multi-particle hopping rate is found, we cannot compare bosonic with fermionic search.

In section 5.4 the influence of graph connectivity on bosonic and fermionic search is investigated. In order to see better how the graph structure influences the runtime, we would again need more data. Moreover, this data should come from a set of graphs where more graphs have a higher average shortest path length, as this data is currently most scarce. Finally, it should be investigated whether the average shortest path length is a good way to quantify graph connectivity at all. Perhaps the algebraic connectivity* or some other measure for the connectivity of a graph is a better predictor of quantum search performance.

An interesting direction for further research would be, investigating the optimal hopping rate for multi-boson or multi-fermion quantum searches, or finding the method of post-measurement marked vertex identification that works best for bosonic or fermionic search.

*The algebraic connectivity of a graph is the second-lowest eigenvalue of the Laplacian of that graph, and is a measure for its connectivity.

A

Analysis of Grover's algorithm

To gain a better understanding of how a quantum search works, we will analyze the single-particle quantum search as defined in section 3.1.2. We will show a way to simplify the dynamics of this quantum search over a complete graph by a reduction of the Hilbert space to a two dimensional subspace. Next, we write the Hamiltonian in the basis of this subspace, and determine its eigenvalues and eigenvectors. Finally, we determine the state $|\psi(t)\rangle$ after evolution under this Hamiltonian, and calculate the probability that the marked vertex will be occupied by the particle as a function of time. This leads to an expression for the optimal time at which to measure the system, which we find scales with $\sqrt{N}$. Thus, we have retrieved Grover's result, that quantum search achieves quadratic speed-up as compared to classical search, illustrating the potential advantage of quantum searches.

A.1. Analysis of Grover's algorithm

For the complete graph, it is known that the optimal value of the hopping parameter, the **critical hopping rate**, is

$$\gamma_c = \frac{1}{N} \tag{A.1}$$

Furthermore, we can write our walking-term more compactly as $-\gamma_c A$, where A is the adjacency matrix of the complete graph. Our Hamiltonian thus becomes

$$\hat{H} = -\gamma_c A - |w\rangle\langle w| \tag{A.2}$$

Our goal is to compute the probability

$$P_w(t) = |\langle w|\psi(t)\rangle|^2 \tag{A.3}$$

and thereby show that the Hamiltonian dynamics draw the particle toward the marked vertex in time scaling as $\sqrt{N}$.

A.2. Reduction to a 2D subspace

In this system there is a symmetry that can be used to simplify the dynamics. Note that, because the graph is complete, all the vertices other than the marked vertex are in a sense equivalent. We can define

$$|w\rangle = |\text{particle on the marked vertex}\rangle \tag{A.4}$$

$$|r\rangle = |\text{particle in a superposition of all other vertices}\rangle = \frac{1}{\sqrt{N-1}} \sum_{i \neq w} |i\rangle \tag{A.5}$$

Then, because all vertices $i \neq w$ are the same, the system stays within the span of $|w\rangle$ and $|r\rangle$.

We would like to compute the Hamiltonian in the basis of these states: the effective Hamiltonian $\hat{H}_{\text{eff}}$. As a step in-between, we determine $A|w\rangle$ and $A|r\rangle$. One can verify that

$$A|w\rangle = \sqrt{N-1}\,|r\rangle\,, \tag{A.6}$$

$$A|r\rangle = \sqrt{N-1}\,|w\rangle + (N-2)\,|r\rangle\,. \tag{A.7}$$

Then the elements in the effective Hamiltonian become

$$\langle w|\hat{H}|w\rangle = -\gamma_c\langle w|A|w\rangle - \langle w|w\rangle\langle w|w\rangle = -\gamma_c\langle w|\left(\sqrt{N-1}\,|r\rangle\right) - 1 = -1 \tag{A.8}$$

$$\langle w|\hat{H}|r\rangle = -\gamma_c\langle w|A|r\rangle - \langle w|w\rangle\langle w|r\rangle = -\gamma_c\langle w|\left(\sqrt{N-1}\,|w\rangle + (N-2)\,|r\rangle\right) = -\gamma_c\sqrt{N-1} \tag{A.9}$$

$$\langle r|\hat{H}|w\rangle = -\gamma_c\langle r|A|w\rangle - \langle r|w\rangle\langle w|w\rangle = -\gamma_c\langle r|\left(\sqrt{N-1}\,|r\rangle\right) = -\gamma_c\sqrt{N-1} \tag{A.10}$$

$$\langle r|\hat{H}|r\rangle = -\gamma_c\langle r|A|r\rangle - \langle r|w\rangle\langle w|r\rangle = -\gamma_c\langle r|\left(\sqrt{N-1}\,|w\rangle + (N-2)\,|r\rangle\right) = -\gamma_c(N-2) \tag{A.11}$$

so that

$$\hat{H}_{\text{eff}} = \begin{pmatrix} -1 & -\gamma_c\sqrt{N-1} \\ -\gamma_c\sqrt{N-1} & -\gamma_c(N-2) \end{pmatrix} \tag{A.12}$$

For large N, $\sqrt{N-1} \approx \sqrt{N}$ and $N-2 \approx N$. Using the optimal $\gamma_c = 1/N$, we get

$$H_{\text{eff}} \approx \begin{pmatrix} -1 & -\frac{1}{\sqrt{N}} \\ -\frac{1}{\sqrt{N}} & -1 \end{pmatrix} \tag{A.13}$$

A.3. Eigenvalues and eigenstates

The eigenvalues of this matrix are

$$E_0 = -1 - \frac{1}{\sqrt{N}} \qquad E_1 = -1 + \frac{1}{\sqrt{N}} \tag{A.14}$$

Thus the energy gap is

$$\Delta E = E_1 - E_0 = \frac{2}{\sqrt{N}} \tag{A.15}$$

The corresponding eigenstates for eigenvalues E_0 and E_1 are respectively

$$|\psi_0\rangle = \frac{|r\rangle + |w\rangle}{\sqrt{2}} \qquad |\psi_1\rangle = \frac{|r\rangle - |w\rangle}{\sqrt{2}} \qquad \Rightarrow \qquad |w\rangle = \frac{|\psi_0\rangle - |\psi_1\rangle}{\sqrt{2}} \tag{A.16}$$

In order to evolve our system, we need to choose an initial state. Since we do not know which vertex is the marked vertex, we cannot choose $|w\rangle$ or $|r\rangle$ as initial state. We can, however, choose the uniform superposition $|s\rangle$ over *all* vertices as initial state:

$$|s\rangle = \frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i\rangle = \frac{1}{\sqrt{N}}\left(|w\rangle + \sqrt{N-1}\,|r\rangle\right) \tag{A.17}$$

In the large-N limit, the $|w\rangle$-component in $|s\rangle$ goes to zero, so we can approximate $|s\rangle \approx |r\rangle$. Then,

$$|\psi_0\rangle \approx \frac{|s\rangle + |w\rangle}{\sqrt{2}} \qquad |\psi_1\rangle \approx \frac{|s\rangle - |w\rangle}{\sqrt{2}} \qquad \Rightarrow \qquad |s\rangle \approx \frac{|\psi_0\rangle + |\psi_1\rangle}{\sqrt{2}} \tag{A.18}$$

A.4. Time evolution and success probability

Using the above formula, the evolved state is

$$|\psi(t)\rangle = e^{-i\hat{H}_{\text{eff}}t}|s\rangle \approx e^{-i\hat{H}_{\text{eff}}t}\left(\frac{|\psi_0\rangle + |\psi_1\rangle}{\sqrt{2}}\right) \approx \frac{1}{\sqrt{2}}\left(e^{-iE_0t}|\psi_0\rangle + e^{-iE_1t}|\psi_1\rangle\right) \tag{A.19}$$

The amplitude on the marked vertex is

$$\begin{aligned}
\langle w|\psi(t)\rangle &\approx \frac{1}{\sqrt{2}}\left(e^{-iE_0t}\langle w|\psi_0\rangle + e^{-iE_1t}\langle w|\psi_1\rangle\right) && \text{(A.20)}\\
&= \frac{1}{\sqrt{2}}\left(e^{-iE_0t}\left(\frac{\langle\psi_0| - \langle\psi_1|}{\sqrt{2}}\right)|\psi_0\rangle + e^{-iE_1t}\left(\frac{\langle\psi_0| - \langle\psi_1|}{\sqrt{2}}\right)|\psi_1\rangle\right) && \text{(A.21)}\\
&= \frac{1}{2}\left(e^{-iE_0t} - e^{-iE_1t}\right) && \text{(A.22)}\\
&= \frac{1}{2}\left(1 - e^{-i\Delta Et}\right) && \text{(A.23)}
\end{aligned}$$

Thus

$$P_w(t) = |\langle w|\psi(t)\rangle|^2 = \left|e^{-\frac{1}{2}i\Delta Et}\cdot\frac{1}{2}\left(e^{\frac{1}{2}i\Delta Et} - e^{-\frac{1}{2}i\Delta Et}\right)\right|^2 = \sin^2\left(\frac{\Delta E\, t}{2}\right) = \sin^2\left(\frac{t}{\sqrt{N}}\right) \tag{A.24}$$

A.5. Runtime and conclusion

We see that the Hamiltonian causes the system to oscillate between $|s\rangle$ and $|w\rangle$. In this sense, the Hamiltonian draws the particle towards the marked vertex in the region $\frac{t}{\sqrt{N}} \in [0, \frac{\pi}{2}]$. The success probability is maximized when

$$\frac{t}{\sqrt{N}} = \frac{\pi}{2} \tag{A.25}$$

i.e.

$$t = \frac{\pi}{2}\sqrt{N} \tag{A.26}$$

Thus, the time it takes this quantum search to find the marked vertex scales as $\Theta(\sqrt{N})$, retrieving Grover's result.

B

The majority vote projector

The success probability $P_r(t)$, the probability that the majority vote method of r rounds performed on the state $|\psi(t)\rangle$ results in a correct guess for the marked vertex, can be determined exactly using a quantum operator. In this section, we construct this operator step by step, and explain how it is used to determine the success probability. Calculating the success probability using a quantum operator enables fast implementation in a computer program. However, this method can only be used for quantum searches over small graphs ($N < 10$) and using a minimal number of particles ($M < 10$).

B.1. The total measurement as measurement of a quantum state

In the following sections we will show that we can calculate the success probability $P_r(t)$ using quantum operators. These quantum operators act on the **total state**

$$\underbrace{|\psi\rangle \otimes |\psi\rangle \otimes \cdots \otimes |\psi\rangle}_{r \text{ times}} = |\psi\rangle^{\otimes r} \tag{B.1}$$

The total measurement, consisting of r measurements of our state $|\psi\rangle$, is in fact one measurement of this total state. To illustrate this, consider a system of one boson walking over a graph of two vertices. A state this system could be in, written as a Fock state, is

$$|\psi\rangle = \frac{\sqrt{3}}{2}|10\rangle + \frac{1}{2}|01\rangle . \tag{B.2}$$

This state is visualized in figure B.1.

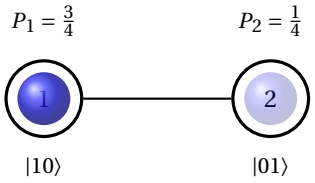


Figure B.1: Representation of the graph of two vertices with a boson (indicated by a blue ball inside each vertex) in a superposition of these vertices. The opacity of the blue ball inside each vertex depends on the probability of finding the boson at that vertex, indicated with P_1 and P_2 for vertices 1 and 2 respectively.

In this system vertex 1 is the marked vertex. Suppose we want to apply two rounds of the majority vote method on this system. The only way in which, after two measurements of $|\psi\rangle$, vertex 1 has the largest total number of particles, is by measuring $|10\rangle$ twice. Thus, the probability that the majority vote method is successful in this case is

$$|\langle\psi|10\rangle|^2 \cdot |\langle\psi|10\rangle|^2 = \frac{3}{4} \cdot \frac{3}{4} = \frac{9}{16} \tag{B.3}$$

Now consider the total state of this system

$$|\psi\rangle \otimes |\psi\rangle = \frac{3}{4}|1010\rangle + \frac{\sqrt{3}}{4}|1001\rangle + \frac{\sqrt{3}}{4}|0110\rangle + \frac{1}{4}|0101\rangle \tag{B.4}$$

Again, only if we would measure $|1010\rangle$ from this state the majority vote would be successful. The probability that this happens is

$$|\langle\psi| \otimes \langle\psi| |1010\rangle|^2 = \frac{9}{16} \tag{B.5}$$

Thus, measuring $|\psi\rangle$ r times is equivalent to measuring the total state $|\psi\rangle^{\otimes r}$ once. For a general system, the total state is a superposition of basis states of the following form

$$|n_1^{(1)} n_2^{(1)} \dots n_N^{(1)} n_1^{(2)} \dots n_n^{(r)}\rangle \tag{B.6}$$

B.2. The majority vote method as quantum operator

The total state will be manipulated using a sequence of quantum operators, all wrapped up into a single **majority vote projector**. The goal is to construct this operator so that it has the following property

$$\hat{M}_r |n_1^{(1)} \dots n_N^{(r)}\rangle = \begin{cases} |n_1^{(1)} \dots n_N^{(r)}\rangle & \text{if the marked vertex has strictly the largest total number of particles,} \\ 0 & \text{otherwise.} \end{cases} \tag{B.7}$$

Here $\hat{M}_r$ is the majority vote projector of r rounds. Applying this operator on a total state $|\psi\rangle^{\otimes r}$, it selects only those basis states $|n_1^{(1)} \dots n_N^{(r)}\rangle$ for which the majority vote method would guess the correct marked vertex. For example, if vertex 1 is our marked vertex, it would only select those basis states $|n_1^{(1)} \dots n_N^{(r)}\rangle$ for which $n_1^{(1)} + \dots + n_1^{(r)} > n_i^{(1)} + \dots + n_i^{(r)}$ for all $i = 2, 3, \dots N$.

Letting $\hat{M}_2$ act on the example total state of before, we get

$$\hat{M}_2 |\psi\rangle \otimes |\psi\rangle = \hat{M}_2 \left(\frac{3}{4}|1010\rangle + \frac{\sqrt{3}}{4}|1001\rangle + \frac{\sqrt{3}}{4}|0110\rangle + \frac{1}{4}|0101\rangle \right) = \frac{3}{4}|1010\rangle \tag{B.8}$$

Thus, the probability that the majority vote method is successful becomes

$$P_2(\psi) = \langle\psi| \otimes \langle\psi| \hat{M}_2 |\psi\rangle \otimes |\psi\rangle \tag{B.9}$$

and for a general r rounds majority vote, we have

$$P_r(\psi) = \langle\psi|^{\otimes r} \hat{M}_r |\psi\rangle^{\otimes r} \tag{B.10}$$

Now, we have reduced the seemingly complex task of calculating the success probability of the majority vote method to a simple quantum operation. What remains is finding the majority vote projector.

B.3. Constructing the majority vote projector

The majority vote projector $\hat{M}_r$ of r rounds will be constructed in four steps.

1. First we would like to find a projector that, for a given vertex i, selects all basis states $|\dots n_i^{(1)} \dots n_i^{(2)} \dots n_i^{(r)} \dots\rangle$ for which

$$\begin{aligned} n_i^{(1)} &= m_1 \\ n_i^{(2)} &= m_2 \\ &\vdots \\ n_i^{(r)} &= m_r \end{aligned} \tag{B.11}$$

   where m_j is the number of particles on vertex i in round j in this basis state. The number of particles on all other vertices $\neq i$ in each round is free: the projector should ignore this. The projector that does this, we will call $\hat{P}[(n_i^{(1)}, \dots, n_i^{(r)}) = (m_1, \dots, m_r)]$, and is constructed as follows

$$\hat{P}[(n_i^{(1)},\ldots,n_i^{(r)}) = (m_1,\ldots,m_r)] = \prod_{j=1}^{r} |m_j\rangle_i^j \langle m_j| \tag{B.12}$$

It consists of a product of $|m_j\rangle_i^j \langle m_j|$-terms. The $\frac{j}{i}$-notation in between the ket and the bra is an abbreviation: written in full it would be $|m_j\rangle_i^{(j)} \langle m_j|_i^{(j)}$, meaning $|m_j\rangle_i^{(j)}$ is a basis state with m_j particles on vertex i in round j (the number of particles on the other vertices left open). Then $|m_j\rangle_i^j \langle m_j|$ becomes an operator acting only on the site in the basis state corresponding to vertex i and round j. As an example, consider again the simple system of before, and the operator $|1\rangle_2^1\langle 1|$. Written in full, this would be $|n_1^{(1)}\ 1\ n_1^{(2)} n_2^{(2)}\rangle \langle n_1^{(1)}\ 1\ n_1^{(2)} n_2^{(2)}|$ (with $n_1^{(1)}, n_1^{(2)}$ and $n_2^{(2)}$ to indicate that the number of particles on these sites is left open), or written differently $I \otimes |1\rangle\langle 1| \otimes I \otimes I$. Each $|m_j\rangle_i^j \langle m_j|$-term in the product selects all basis states with m_j particles at vertex i in round j from the superposition. The product of all $|m_j\rangle_i^j \langle m_j|$-terms selects all basis states for which the constraints of (B.11) are satisfied. This is because if one of the constraints would be violated, say $n_i^{(j)} \neq m_j$, then the term $|m_j\rangle_i^j \langle m_j|$ acting on the basis state $|\ldots n_i^{(1)} \ldots n_i^{(2)} \ldots n_i^{(r)} \ldots\rangle$ would evaluate to 0, making the entire product equal to 0. Thus we have found our first projector.

2. As a second step, we would like to find a projector that selects all basis states from $|\psi\rangle^{\otimes r}$ for which the total number of particles on vertex i is m. Thus, we should construct a projector that selects all basis states $|\ldots n_i^{(1)} \ldots n_i^{(2)} \ldots n_i^{(r)} \ldots\rangle$ for which $n_i^{(1)} + \cdots + n_i^{(r)} = m$. This projector, written as $\hat{P}[n_i^{\text{tot}} = m]$, is given by

$$\hat{P}[n_i^{\text{tot}} = m] = \sum_{m_1+\cdots+m_r=m} \hat{P}[(n_i^{(1)},\ldots,n_i^{(r)}) = (m_1,\ldots,m_r)] \tag{B.13}$$

Each projector $\hat{P}[(n_i^{(1)},\ldots,n_i^{(r)}) = (m_1,\ldots,m_r)]$ in the sum selects all basis states such that $(n_i^{(1)},\ldots,n_i^{(r)}) = (m_1,\ldots,m_r)$. By summing over all possible ways to have $m_1 + \cdots + m_r = m$, we get all basis states that have a total of m particles on vertex i.

3. The following step is creating a projector $\hat{P}[n_w^{\text{tot}} > n_i^{\text{tot}}]$ that selects all basis states for which $n_w^{\text{tot}} > n_i^{\text{tot}}$, where w is the marked vertex and i is a vertex of choice. To do this, consider the product of

$$\hat{P}[n_w^{\text{tot}} = m]\hat{P}[n_i^{\text{tot}} = l] \tag{B.14}$$

This is an operator selecting basis states that have both m particles in total on the marked vertex, and l particles in total on vertex i, because if either $\hat{P}[n_w^{\text{tot}} = m]$ or $\hat{P}[n_i^{\text{tot}} = l]$ is not satisfied (returns 0 when acting on the basis state), the product becomes 0 as well. Now, for a general state $|\psi\rangle^{\otimes r}$, m can be at most Mr, as in each round vertex w can be occupied by at most M particles. Thus we are interested in selecting all basis states for which $n_w^{\text{tot}} = 1$ and $n_i^{\text{tot}} \leq 0$, or $n_w^{\text{tot}} = 2$ and $n_i^{\text{tot}} \leq 1$, or … $n_w^{\text{tot}} = Mr$ and $n_i^{\text{tot}} \leq Mr - 1$. The projector of interest therefore is

$$\hat{P}[n_w^{\text{tot}} > n_i^{\text{tot}}] = \sum_{m=0}^{Mr} \sum_{l=0}^{m-1} \hat{P}[n_w^{\text{tot}} = m]\hat{P}[n_i^{\text{tot}} = l] \tag{B.15}$$

4. Finally, we would like to select all basis states for which the inequality $n_w^{\text{tot}} > n_i^{\text{tot}}$ holds for all $i \neq w$, as only in this case would vertex w be guessed as the marked vertex by the majority vote method. More concretely, we would like $n_w^{\text{tot}} > n_1^{\text{tot}}$ and $n_w^{\text{tot}} > n_2^{\text{tot}}$ and … and $n_w^{\text{tot}} > n_N^{\text{tot}}$ to be true (skipping $n_w^{\text{tot}} > n_w^{\text{tot}}$ of course). This results in our final projector: the majority vote projector

$$\hat{M}_r = \prod_{i \neq w} \hat{P}[n_w^{\text{tot}} > n_i^{\text{tot}}] \tag{B.16}$$

Bibliography

[1] Shantanav Chakraborty, Leonardo Novo, and Jérémie Roland. On the optimality of spatial search by continuous-time quantum walk. *Physical Review A*, 102(3):032214, September 2020. ISSN 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.102.032214. URL http://arxiv.org/abs/2004.12686. arXiv:2004.12686 [quant-ph].

[2] Andrew M. Childs and Jeffrey Goldstone. Spatial search by quantum walk. *Physical Review A*, 70(2): 022314, August 2004. ISSN 1050-2947, 1094-1622. doi: 10.1103/PhysRevA.70.022314. URL http://arxiv.org/abs/quant-ph/0306054. arXiv:quant-ph/0306054.

[3] Edward Farhi and Sam Gutmann. An Analog Analogue of a Digital Quantum Computation, December 1996. URL http://arxiv.org/abs/quant-ph/9612026. arXiv:quant-ph/9612026.

[4] Lov K. Grover. A fast quantum mechanical algorithm for database search, November 1996. URL http://arxiv.org/abs/quant-ph/9605043. arXiv:quant-ph/9605043.

[5] Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. ISSN 0162-1459. doi: 10.2307/2282952. URL https://www.jstor.org/stable/2282952. Publisher: [American Statistical Association, Taylor & Francis, Ltd.].

[6] Aravind Karthigeyan. Quantum Search with Non-Interacting Bosons and Fermions.

[7] J. M. Leinaas and J. Myrheim. On the theory of identical particles. *Il Nuovo Cimento B (1971-1996)*, 37 (1):1–23, January 1977. ISSN 1826-9877. doi: 10.1007/BF02727953. URL https://doi.org/10.1007/BF02727953.

[8] C Li, H Wang, W De Haan, C J Stam, and P Van Mieghem. The correlation of metrics in complex networks with applications in functional brain networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(11):P11018, November 2011. ISSN 1742-5468. doi: 10.1088/1742-5468/2011/11/P11018. URL https://iopscience.iop.org/article/10.1088/1742-5468/2011/11/P11018.

[9] Tijmen van Gelder. How small c has to be, January 2026. URL https://www.youtube.com/watch?v=Go2UeT3eIsI.