Quantum Algorithm for Linear Systems of Equations

Alfred Nguyen
Department of Computer Science
Technical University of Munich
05748 Garching, Bavaria
alfred.nguyen@outlook.com

***Abstract*—Linear systems are a fundamental problem in math or can be found in subroutines in more complex tasks. Linear systems are in the form of $A\vec{x} = \vec{b}$, where A is a given matrix, $\vec{b}$ is a given vector and $\vec{x}$ is the unknown to be solved. The HHL (Harrow, Hassidim, and Lloyd) algorithm is a quantum algorithm, that is able to solve these linear systems of equations exponentially faster than its classical counter part. Though, there are a few caveats to consider. We assume that we only want to solve for an expectation value of some operator on $\vec{x}$, e.g. $\vec{x}^{\dagger} M \vec{x}$ for some matrix M. That means we are not interested in the whole solution of $\vec{x}$. Also, we assume that the matrix A is sparse and is in the size of $N \times N$. Given these requirements, classical algorithms can solve this problem in $\mathcal{O}(N)$, whereas the HHL algorithm can solve this problem in $\mathcal{O}(log(N))$. This gives us an exponential speedup over the classical method.**

***Index Terms*—Harrow-Hassidim-Lloyd (HHL) quantum algorithm, Quantum Fourier transform (QFT), Inverse Quantum Fourier transform (IQFT), Quantum Phase Estimation (QPE), is that everything?**

I. Introduction

Quantum computing has a lot of potential in many various domains. It leverages the power of quantum superposition and entanglement to perform computations that a classical compuer cannot similate. In some cases, quantum computer algorithms can provide an exponential speed up in comparison to common classical methods. The most famous case being the Shor's algorithm for factoring large numbers, possibly cracking our current RSA encryption.

Solving linear systems is a fundamental subproblem in many scientific, engineering and mathematical discplines. Currently, the demand for processing data sets is increasing. The sizes of these datasets which make up the equations, is growing well into terabytes and petabytes of data. Obtaining solutions for these kind of problems can be very computational expensive. Even an approximation of these solutions with N unknowns take at least N timesteps. Just outputting the solution takes at least N steps, as we have to read out all the N unkowns for the whole solution.

Though, oftentimes we are not interestd in the full solution vector. For example, we can assume that we are only interested in a function, that determines the weights of some subsets in a specific region of the vector. Then we can achieve an exponential faster approximation by using quantum computers. In the end we will observe, that the sparsity and conditioning of the problem matrices A, have comparable dependency on the runtime, but the error dependency is exponentially worse.

Especially in quantum machine learning these kind of problems arise very frequently. Using large datasets finding patterns, classifying and clustering are very common methods which demand a lot of resources. Many of the quantum machine learning algorithms extend or use the following algorithm as a subroutine.

This paper will focus on an algorithm, which can estimate features of the solution vector of a linear system of equations. The HHL algorithm, designed by Aram Harrow, Avinatan Hassidim and Seth Lloyd, was introduced in 2009. It is one of the groundbreaking algorithms in quantum computing alongside Shor's factoring algorithm, Grover's search algorithm, and the quantum fourier transform (QFT). Given an $N \times N$ sparse matrix A, the HHL algorithm runs in $\mathcal{O}(log(N))$, whilst the classical counterpart runs in $\mathcal{O}(N)$. This promises us an exponential speed up over the commonly used classical method.

1) Outline: Firstly, this paper will give a rough overview of the HHL algorithm, specifying the problem in detail, giving a mathematical idea of the algorithm, followed by the explanation of the quantum circuit. Then, we will discuss the HHL algorithm, going through all the phases of the algorithm. Next, we evaluate and analyze its runtime by comparing the HHL algorithm by its classical counterpart. Afterwards, we will take an outlook at future works, exploring potential applications, variations of the HHL algorithm and will relate it to IT security. In the end we will give a final conclusion, summarizing our findings.

II. Overview of the algorihtm

The following section will describe how the algorithm works in general. Firstly, it will specify the problem statement. After that, it will describe a short mathematical explanation. Then, it will sketch out the steps of the algorithm.

A. Problem statement

Given an NxN hamiltonian matrix A and $\vec{a}$ vector $\vec{b}$, we want to solve for the vector $\vec{x}$ such that,

$$A\vec{x} = \vec{b} \tag{1}$$

To solve for x the equation can be rewritten as

$$\vec{x} = A^{-1}\vec{b} \tag{2}$$

Note that to encode $\vec{b}$ we need $\mathcal{O}(log_2 N)$ qubits.

As described earlier the hermitian matrix A^{-1} can be split into its spectral decomposition. A can be represented in terms of its eigenvalues $U_1 ... U_n$ and eigenvectors $\lambda_1 ... \lambda_n$.

$$A = UDU^\dagger = \begin{pmatrix} U_1 & ... & U_n \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n \end{pmatrix} \begin{pmatrix} U_1^\dagger \\ ... \\ U_n^\dagger \end{pmatrix} \tag{3}$$

This means, if we can find the eigenvalues and eigenvectors of A we can then solve the linear equation quite easily. Classical solutions involving spectral decomposition are not faster than other standard algorithms such as Gaussian Elimination. Though, estimating eigenvalues and eigenwerte can be performed quite efficiently by quantum methods. Via amplitude amplification, QPE can be accelerated to generate the eigenvalues and eigenvectors in $\mathcal{O}(log_2 N)$ steps.

In the quantum version the linear equation looks like this

$$|x\rangle = A^{-1} |b\rangle \tag{4}$$

where $|b\rangle$ and $|x\rangle$ are the quantum state of the $\vec{b}$ and $\vec{x}$ vectors respectively. Note that $|x\rangle$ is just a quantum state. We can not read every element to achieve the vector $\vec{x}$. So far we were able to perform everything in $\mathcal{O}(log_2 N)$. Reading out every entry of $\vec{x}$ would take $\mathcal{O}(N)$ steps which would destroy our speed up. But, as we are only interested in an approximation, we can compute an expectation value $\langle x| M |x\rangle$, where M is some linear operator. With this method we can extract many statistical features like normalization, distribution of weights, moments, etc.

B. Mathematical Overview

We will now look at a mathematical overview of what is happening in the quantum circuit. We will also assume that the matrix A is hermitian. If A is not hermitian, we can write A as a hermitian like this

$$A^\dagger = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \tag{5}$$

As already mentioned the matrix can now be described as a linear combination of its outer products of its eigenvectors and its eigenvalues. In the quantum version the formula looks like this

$$A = \sum_{i=0}^{N-1} \lambda_i |u_i\rangle \langle u_i| \tag{6}$$

where $|u\rangle_i \langle u|_i$ are the outer products of A and λ_i are the eigenvalues of A. N is the size of the matrix A. We can rewrite the formula for the inverse A^{-1} as such

$$A^{-1} = \sum_{i=0}^{N-1} \lambda_i^{-1} |u_i\rangle \langle u_i| \tag{7}$$

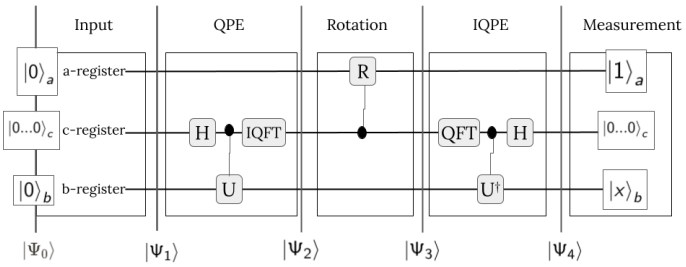

Fig. 1. Example Circuit

Similary $|b\rangle$ can be expressed in the eigenbasis of A like this

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle \tag{8}$$

We now have all the tools to solve the equation by inserting the definition of A^{-1} and $|b\rangle$ into our original equation,

$$\begin{aligned} |x\rangle &= A^{-1} |b\rangle \\ &= \left(\sum_{i=0}^{N-1} \lambda_i^{-1} |u_i\rangle \langle u_i| \right) \left(\sum_{j=0}^{N-1} b_j |u_j\rangle \right) \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \lambda_i^{N-1} |u_i\rangle \langle u_i| b_j |u_j\rangle \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \lambda_i^{-1} b_j |u_i\rangle \langle u_i|u_j\rangle \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \lambda_i^{-1} b_j |u_i\rangle \delta_{ij} \\ &= \sum_{i=0}^{N-1} \lambda_i^{-1} b_j |u_j\rangle \end{aligned}$$

As we can see we can encode the solution $|x\rangle$ of our linear system into our quantum system quite nicely. That means, $|x\rangle$ can be calculated, only by determinining the eigenvectors and eigenvalues of A. Using Quantum Phase Estimation, calculating the eigenvalues and eigenvectors for a unitary U can be very efficient.

C. Quantum Circuit

We will now look at how the algorithm is implemented in the quantum circuit. We will firstly look at the sets of qubits need for the algorithm. Then, we will describe the phases of the procedure.

1) Registers: Fig. 1 shows the scheme of a simple quantum circuit for the HHL algorithm. We have three registers which describe three different sets of qubits in the quantum circuit.

The a-register contains the ancilla qubit. It is used for the inversion of the eigenvalues and will be explained in detail later on.

The c-register, oftentimes refered to as the clock-register, is used for the quantum phase estimation part. It is related to the time (clock) of the controlled rotation of the qubits and will store the eigenvalues after performing QPE.

The b-register contains the $\vec{b}$ vector which is encoded into a quantum state $|b\rangle$. After the whole HHL procedure is done the b-register will contain the solution state $|x\rangle$.

2) Phases: The procedure of the quantum circuit can be split into 5 phases:

- State preparation
- Quantum phase estimation (QPE)
- Inversion of eigenvalues
- Inverse quantum phase estimation (IQPE)
- Measurement of $|x\rangle$

In the state preparation phase, the vector $\vec{b}$ will be encoded into a quantum state $|b\rangle$ and the A matrix will be encoded as a hamiltonian, which is a unitary operator $U = e^{iAt}$ into the QPE and IQPE operations.

The Quantum Phase Estimation will then calculate the eigenvalues and eigenvectors of the A matrix.

Then, we perform the rotation and invert the eigenvalues through rotary operations. Unfortunately, these operations have a probability to fail, as they are not unitary operators. The ancilla will detect wether the rotation was successfull or not. It will either collapse to $|0\rangle$ or $|1\rangle$ for failure and success respectively. If the rotation is not successfull, the procedure has to be repeated from the beginning. If the rotation was successfull we can continue the procedure. The problem is, that the qubits in the b-register and c-register are entangled. This means that we cannot factorize the result into a tensor product of the c-register and b-register. As a result, we cannot convert the b-register into the $|0\rangle$ / $|1\rangle$ measurement basis with the desired amplitudes. We will need to uncompute the state so that it gives the right results in the $|0\rangle$ / $|1\rangle$ measurement during which the b-register and c-register will be unentangled. That means we have undo all operations until now to unentangle the states, keeping the inverted eigenvalues though.

This is achieved by the Inverse Quantum Phase Estimation which undos all steps we performed in the QPE phase, leaving us with the a $|0...0\rangle$ state in the c-register and the $|x\rangle$ state in the b-register.

Lastly, the $|x\rangle$ state is to be measured. As mentioned earlier, we can only read out an approximation of an expectation value $\langle x| M |x\rangle$.

III. The HHL Algorithm

The next section will give a more detailed walkthrough of the HHL algorithm. Thereby, it will go through all the 5 phases namely, state preperaration, quantum phase estimation, inversion of eigenvalues, inverse quantum phase estimation and lastly the measurement.

A. State Preparation

In total we have $n_b + n + 1$ qubits. In the beginning, they are all initialized in their zero state as

$$\begin{aligned} |\Psi_0\rangle &= |0\dots0\rangle_b \ |0\dots0\rangle_c \ |0\rangle_a \\ &= |0\rangle_b^{\otimes n_b} \ |0\rangle_c^{\otimes b} \ |0\rangle_a \end{aligned} \tag{9}$$

We now have to load the vector $\vec{b}$ into the b-register. This is achieved by amplitude encoding

$$\vec{b} = \begin{pmatrix} b_0 \\ \vdots \\ b_{N-1} \end{pmatrix} \Leftrightarrow |b\rangle = \sum_{i=0}^{N-1} b_i |i\rangle \tag{10}$$

Thus, $|b\rangle$ is loaded into the b-register as such

$$|\Psi_1\rangle = |b\rangle_b \ |0\dots0\rangle_c \ |0\rangle_a \tag{11}$$

We have successfully encoded the $\vec{b}$ into our b-register. We now continue with the QPE.

B. Quantum Phase Estimation

We will only briefly go through the specifics of the QPE and will not discuss each step in detail, as this is not the main topic of this paper. For further explanations refer to this paper. Todo

Inser paper here

As already mentioned, the QPE is a procedure to evalute an estimate of eigenvalues. It consists of three phases namely, creating superpositions of the clock-bits via Hadamard gates, controlled rotation via the unitary U and the IQFT. After QPE we will have an estimate of the eigenvalues of the unitary U. As we have encode A as as a Hamiltonian $U = e^{iAt}$, the phase of the eigenvalue of U is proportional to the eigenvalue of A. We have to define a scaled version of our eigenvalues λ_j.

$$\widetilde{\lambda_j} = \frac{N\lambda_j t}{2\pi} \tag{12}$$

where t can be choosen freely so that the scaled eigenvalues $\widetilde{\lambda_j}$ are integers.

Thus, the eigenvalues of A will be stored in the c-register after QPE as

$$\begin{aligned} |\Psi_2\rangle &= |b\rangle_b |\widetilde{\lambda}\rangle_c |0\rangle_a \\ &= \sum_{j=0}^{N-1} b_j |u_j\rangle_b |\widetilde{\lambda_j}\rangle_c |0\rangle_a \end{aligned} \tag{13}$$

Notice, that the b-register is in the same form as we have discussed in the previous section. The b-register represents the $|b\rangle$ state in the eigenbasis $|u_j\rangle$ of A like this

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle \tag{14}$$

Now that we have achieved the eigenvalues and have represented our $|b\rangle$ in the eigenbasis of A, we invert the eigenvalues u_i.

C. Inversion of the eigenvalues

In the next step we invert the eigenvalues. This is achieved by the rotation of the ancilla qubit in the a-register by the eigenvalues in the c-register. The state of the registers after the rotation looks like this

$$|\Psi_3\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle_b |\widetilde{\lambda_j}\rangle_c \left(\sqrt{1 - \frac{C^2}{\widetilde{\lambda_j^2}}} |0\rangle_a + \frac{C}{\widetilde{\lambda_j}} |1\rangle_a \right) \tag{15}$$

where C is a constant that should be chosen to be as large as possible to increase the success probability. Currently, the state of the a-register can either collapse into $|1\rangle$ or $|0\rangle$. The probability of the a-register collapsing to $|1\rangle$ is $\left|\frac{C}{\widetilde{\lambda_j}}\right|^2$. If the ancilla qubit collapses to $|0\rangle$, the whole procedure has to be repeated from the beginning. As mentioned earlier this has to be done, because the rotation process is not unitary and has a probability to fail.

We assume that our rotation process was successfull and the ancilla bit collapses to $|1\rangle$. The registers will look as such

$$|\Psi_4\rangle = \frac{1}{\sqrt{\sum_{j=0}^{N-1} \left|\frac{b_j C}{\widetilde{\lambda_j}}\right|^2}} \sum_{j=0}^{N-1} b_j |u_j\rangle_b |\widetilde{\lambda_j}\rangle_c \frac{C}{\widetilde{\lambda_j}} |1\rangle_a \tag{16}$$

Note that term in front of the sum is just a factor to normalize the state. Lets call this normalization factor D. We see that we now have a term $\frac{C}{\widetilde{\lambda_j}}$ that represents the inverted eigenvalues. This term can be moved freely as it is just a scalar, such that the scalar is applied to the b-register

$$|\Psi_4\rangle = D \sum_{j=0}^{N-1} \frac{C}{\widetilde{\lambda_j}} b_j |u_j\rangle_b |\widetilde{\lambda_j}\rangle_c |1\rangle_a \tag{17}$$

If we go back to our idea from the mathematical overview section, our b-register is in the same form as our solution state

$$|x\rangle = \sum_{i=0}^{N-1} \lambda_i^{-1} b_j |u_j\rangle \tag{18}$$

That means our solution is already encoded in our registers. The problem here is, that we cannot read the solution out yet. This has to with the states in the b-register and c-register being entangled with each other. That means, we cannot convert the b-register into a $|0\rangle / |1\rangle$ measurement. In the following, we have to undo all operations to unentangle the state in the b-register and c-register, to achieve the correct result.

D. Inverse Quantum Phase estimation.

The unentangling of the registers is achieved throught the IQPE which backtracks all calculations of the QPE. We are left with the following state

$$\begin{aligned} |\Psi_5\rangle &= \frac{1}{\sqrt{\sum_{j=0}^{N-1} \left|\frac{b_j C}{\widetilde{\lambda_j}}\right|^2}} \sum_{j=0}^{N-1} \frac{C}{\lambda_j} b_j |u_j\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \\ &= \frac{C}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left|\frac{b_j C}{\lambda_j}\right|^2}} |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \end{aligned} \tag{19}$$

The a-register is untouched and still holds the $|1\rangle$ state as before. The c-register however is reset to the zero state $|0\rangle_c^{\otimes n}$ as in the beginning of the process at $|\Psi_0\rangle$. It is now unentangled from the b-register. The b-register now contains the solution $|x\rangle$ after the uncomputation and can be measuremed correctly.

We can furthermore simplify the term as we assume that C is real and that the eigenvectors $|u_i\rangle$, and the $|b\rangle$ state are normalized. Then we can simplify to

$$\begin{aligned} |\Psi_5\rangle &= \frac{1}{\sqrt{\sum_{j=0}^{N-1} \left|\frac{b_j}{\lambda_j}\right|^2}} |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \\ &= |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \end{aligned} \tag{20}$$

We can now read out the result $|x\rangle$ in the b-register.

E. Measurement

As mentioned earlier, we cannot obtain the whole solution for the $\vec{x}$ as reading out all the entries would cost us $\mathcal{O}(N)$ steps. This would ommit our speedup of $\mathcal{O}(logN)$. That means, by measuring we will only obtain an estimate of specific features of $|x\rangle$. Using a linear operator M we can perform various measurements on $|x\rangle$ by calculating the inner product as such

$$\langle x| M |x\rangle \tag{21}$$

With this we can extract various statistical features of $|x\rangle$ like the norm of the vector, the average of the weight of the components, moments, probability distributions, localization and concentration in specific regions, etc.

IV. Evaluation

In this section, we evaluate the performance of the HHL algorithm and will shortly look at resource requirements.

The most common known method to solve a system of linear equations is Gaussian Elimination. The time complexity of the Gaussian is $\mathcal{O}(N^3)$, which is drastically slower than other classical methods. As we are only interested in an estimate of an expectation value $|x\rangle M |x\rangle$, we can take a look of similar classical methods, considering other constraints.

In the following we will compare the HHL algorithm, to the classical conjugate gradient descent algorithm, focusing on time complexity and other factors that impact its efficiency.

A. Conjugate Gradient Descent

The classical conjugate gradient descent algorithm is very common method for solving linear systems of equations. It uses the conjugate direction method, which can locate the minimum of a function. By iteratively looking for solution vectors, the procedure converges towareds the solution of the linear system. This will provide us a suitable benchmark for analyzing the efficiency of the HHL algorihtm. Not only is it one of the fastest classical algorithms, it has very similar constraints set and calculates similar results (eg. $\vec{x}^\dagger M\vec{x}$).

B. Comparison of Time Complexity

TABLE I
TIME COMPLEXITY COMPARISON

Conjugate Gradien Descent	HHL Algorithm
$\mathcal{O}(\kappa s \log\left(\frac{1}{\epsilon}\right) N)$	$\mathcal{O}\left(\frac{\kappa^2 s^2}{\epsilon} \log N\right)$
$\Rightarrow \mathcal{O}(N)$	$\Rightarrow \mathcal{O}(log(N))$

As shown in the table, the HHL algorithm offers an exponential improvement compared to the conjugate gradient descent method. Conjugate gradient descent. Conjugate gradient descrent achieves a time complexity of $\mathcal{O}(N)$, whereas the HHL algorihtm runs in $\mathcal{O}(log(N))$.

Now we will take a more detailed look a the other factors involved in the timecomplexity, where:

- s is the s-sparsness
- κ is the condition number
- ϵ is the accuracy

The sparsness s, describes to number of non-zero entries per row (e.g. a 2-sparse matrix only contains 2 non-zero entries per row). The condition number κ, describes how sensitiv the output of a function is on the error of the input. That means, a function is well conditioned if the output of a function does not change a lot for bigger errors in the input. These two factors are related to the QPE phase as, a unitary e^{iAt} has to be generated. This step heavily relies on the sparsity s and conditioning κ of the matrix A to be performed efficiently. If not, this process would grow $\mathcal{O}(N^c)$ for some constant c. Again, this would ommit our speedup of $\mathcal{O}(log(N))$.

Lastly, ϵ describes the accuracy of our desired solution. We can observer, that the sparsity s and condition number κ are quadratic in the HHL algorihtm, whereas in the conjugate gradient descent algorithm, both factors operate linearly. Furthermore, the accuracy ϵ is exponentially worse in the HHL algorithm. Thus, in terms of sparsity, accuracy and and condition number, the HHL algorithm a worse runtime over the conjugate gradient descent algorithm. If these prefactors where not worse than in the classical solution, this would have bizarre implications. One can show that if the dependency is better or equal than in the classical solution, one could solve *NP-Complete* problems in exponential faster times on quatum computers, which seems unlikely. All in all, these constraints are important to consider when choosing applications, as they have great effect on the runtime.

C. Resource requirements

The resource requirements to implement the HHL algorithm are substantial. Looking at the structure of the algorithm is has very similar structure to the Shor's algorithm. We can use Shor's procedure as a lower bound to estimate the resource requirements. Shor's algorithm for 2048 RSA, needs roughly 4000 logical qubits to work properly. To compensate for error correction and other quantum effects in the circuit, this would take millions of qubits. Although, these numbers have been dropped many times over the past couple of years.

V. OUTLOOK

In this section we will take a look at the future outlooks of the HHL algorithm We will explore potential applications given the constraints set of the HHL algorithm. Then we will delve into different variations and optimizations which can improve the performance. Lastly, we will discuss some broad perspective, that the HHL algorithm offers in the field of quantum computing.

A. Applications

The main problem wih the HHL algorithm is, that it does not output the full solution vector, but rather the solutino state $|x\rangle$. Not to be forgotten, is that the matrix A has to be sparse for the HHL algorithm to work efficiently. But in some cases these constraints are not a problem and can therefore be solved by the HHL algorithm. We will now discuss some examples that utilize the algorithm.

1) Analyzing Large Sparse Electrical Networks: The HHL algorithm can be applied to analyze large sparse electrical networks, where we are interested in an estimate of a specific feature. Electrical networks consist of vast numbers of interconnected components, like generators, transformers, transmission lines, etc. Though, these networks have a high number of components, they have a relative low amount of connections between the each other. Thus, systems can be modeld as a graph which correspond to large sparse matrices. By being able to calculate the inverse of a matrix A^{-1} efficienly, we can deduct many network properties, such as resistance. This can help us to further optimize the electrical powersystem. Conventionally, we would have to rely on iterative algorithms, which are as we saw earlier, exponentially slower in this specific case.

2) Machine Learning (Least-Square Estimation): In the field of machine learning the HHL algorithm can also help improve calculations. The HHL algorithm also plays a significant role in the field of machine learning, as it can improve computational intensive calculations. For example in least-square estimation, the HHL algorithm can be used to estimates of inverse matrices. This can be very helpful in data fitting tasks, enabling efficient parameter estimation and model optimization.

All in all, finding more applications for the HHL algorithm is crucial, given its specific constraints and promising capabilities.

B. Variations and Optimizations

Current research and developments have been able to find variations and optimizations to the HHL algorithm. These changes have brought more efficient ways to perform the HHL algorithm and expand its applicability. We will now discuss some examples of improvements done to the HHl algorithm.

1) Ancilla Bit Requirement: Certain variants allow the HHL algorithm to operate without an ancilla bit. This eliminates the probability of failure in the rotation step of the eigenvalues, which leads to a faster walkthrough, as one must not repeat the process mutliple times at failure. Additionally, this increases the reliability of the algorithm.

2) QRAM: Quantum Random Access Memory (QRAM) is the quantum computer equivalent of the classical ram in a computer. It enables the the quantum computer to directly access quantum states for computation without having to encode the input. This would provide us with a very efficient way to load our state $|b\rangle$ into our quantum circuit.

Exploring these variations and identifying novel optimizations can contribute to the advancement and practical implementation of the HHL algorithm.

C. IT Security

Although, the HHL algorithm is primarily designed for solving linear system, there are potential applications in the field of IT security. We will now discuss some examples how one could utilize the algorithm IT security.

1) Solving Large-scale Linear Systems: The HHL algorithm can contribute to the efficient calculation of large-scale linear systems. These are required in many IT security protocols, such as secure multi-party computation or zero-knowledge proofs.

Multi-party computation is as protocol that enables multiple different machines to computer a function, whilst not revealing their data to each other. The protocol achieves this by using linear systems as a subroutine to achieve secrecy.

Zero knowledge proofs are methods that enable a party to demonstrate certain features to another party without revealing any other information except the validity of that feature. This can be usefull to verify your identity without giving out any personal data. Oftentimes these zero knowledge proofs can be modeled in linear systems, which then can be processed by the HHL algorithm.

2) Pattern recognition for fraud dection: Pattern recognition for fraud detection is also a possible application for the HHL algorithm. Again, this plays in the field of machine learning. Here various big data analysis methods are used to detect fraudulant activites, which can be accelerated by the HHL algorithm.

D. Perspectives

The HHL algorithm has made significant contributions, especially in the field of quantum machine learning. Though, there are no ground breaking applications so far, like the Shor's algorithm that has the ability to break the RSA encryption. Nonetheless, the ongoing discussions and research on the algorithm enable us to discover new optimizations and applications. The advances in the quantum algorithms, including the HHL algorithm, have laid the foundation for future developments in the field. But because of its generality of the HHL algorithm, it has the potential to be used as a subroutine in various domains.

While experiments have demonstrated that the HHL algorithm works on a small quantum computers with high fidelity, more advancements have to be done in the hardware. Better error correction and scaling is need to unlock the full potential of quantum computing