Empirical evaluation of a quantum accelerated approach for the central path method in linear programming*

Vijay Adoni[1,*,†], Sajad Fathi Hafshejani[1,†] and Daya Gaur[1,†]

[1]Department of Math and Computer Science, University of Lethbridge, Lethbridge, AB, Canada

Abstract

The central path method is a crucial technique for solving a wide range of optimization problems. The method relies on an equation solving step, which hits its limit for very large instances in practise. This paper proposes to explore the use of quantum approaches to enhance the central path method's performance when solving very large linear programs. We will go through the potential benefits and limitations of replacing the iterative equation-solving step with the HHL quantum algorithm. We experiment with several instance types using each proposed algorithm and evaluate their effectiveness through numerical simulations to find a promising approach for the central path method.

Keywords

Quantum Algorithms, Central Path Method, Interior Point Methods.

1. Introduction

Linear programming (LP) is a crucial tool in both theoretical and practical science and engineering domains. It has been extensively used to solve optimization problems in various areas, including operations research, engineering, economics, or even in more abstract mathematical areas such as combinatorics. LP has applications in machine learning and numerical optimization. Some of the examples of application of LP are $\mathscr{L}_1$-regularized support vector machines (SVMs) [1], basis pursuit (BP) problems [2], sparse inverse covariance matrix estimation (SICE) [3], non-negative matrix factorization (NMF) [4], MAP inference [5], and adversarial deep learning [6, 7]. Fung et al. [8] introduced a technique for learning a kernel function that is a linear combination of other positive semi-definite kernels. They demonstrated how diagonal dominance constraints can be utilized to obtain an approximate kernel through linear programming. This method can be employed for feature selection using a blend of kernels. This is an important use of linear programming in computing Kernels for support vector machines. The results mentioned in this paragraph are illustrative of the utility of linear programming in solving optimization problems arising in machine learning and data science.

When it comes to solving linear problems, there are a variety of techniques available [9]. Depending on the specifics of the problem at hand, different methods may be more appropriate than others. Factors like problem size and structure, desired level of accuracy, and computational efficiency all come into play when determining which method to use. Regardless of which technique is employed, the ultimate goal is to identify the feasible solution space and then optimize the objective function within that space efficiently. LP problems can be complex and time-consuming to solve strictly, so n-approximation algorithms are used to determine the proximity of the solution to optimal. Another way to reduce the time it takes to solve an LP is to use quantum computations for the expensive steps instead of the traditional classical algorithm, resulting in a possible speed-up, the object to study in this paper.

Quantum computing is a new and exciting way to process information that utilizes the principles of quantum mechanics. Unlike traditional computing, which uses bits to represent data and relies on binary logic, quantum computing uses qubits that can be in a superposition of states possibly entangled. In November 2022, IBM created the largest quantum computer, Osprey, with 433 qubits capable of holding 2^{433} states simultaneously. Ongoing research is making quantum computing practical for solving certain problems in optimization and simulation, which are difficult or impossible with classical methods. As the field continues to develop and advances are made in hardware and software, quantum methods are becoming more practical for a wide variety of applications. One of the most promising quantum methods for solving linear equations is the HHL algorithm [10], created by Harrow, Hassidim, and Lloyd, which offers an exponential speedup compared to classical algorithms

Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — International Workshop on Quantum Data Science and Management (QDSM'23), August 28 - September 1, 2023, Vancouver, Canada

*Corresponding author.

†These authors contributed equally.

vijay.adoni@uleth.ca (V. Adoni); sajad.fathihafshejan@uleth.ca (S. F. Hafshejani); daya.gaur@uleth.ca (D. Gaur)

https://gaurdr.github.io (D. Gaur)

0000-0002-5731-8234 (S. F. Hafshejani); 0000-0001-6876-6000 (D. Gaur)

under certain critical assumptions. This algorithm is useful for solving sparse systems of linear equations, where the number of non-zero entries in the matrix is much smaller than the total number of entries.

The following section will review related work in this area. It will also examine the evolution of the methods used to solve LP problems and a few quantum computing techniques needed for solving linear equations.

This review will provide a foundation for our proposed approach and help to establish the significance of our work in this field described in Section 3.

2. Related Work

Over the years, the complexity of solving linear programs has substantially improved due to algorithm design techniques and computational hardware advancements. The key developments that have played a crucial role in this improvement will be discussed in this section. Algorithms used for solving linear optimization programs date back to 1939 with the introduction of the graphical method. However, the simplex method, introduced by Dantzig [11], has become a more prominent and frequently used method. Karmarkar [12] proposed interior point methods (IPMs) and showed that they could solve linear programs much faster than the simplex method in the worst-case. This led to extensive research on interior point methods, resulting in the development of many other algorithms since then.

The existence of an interior path, known as the central path, that converges to an optimal solution paved the way for efficient algorithms that find the optimal solution quicker than the simplex method. There have been several successful attempts at traversing this central path. Methods where the solutions lie in the interior of the feasible region at each iteration while maintaining feasibility at each step are known as feasible interior point methods (see Roos et al. [13]). Methods where the solutions are strictly positive, with an initial solution outside the feasible region, and which converge to an optimal are known as infeasible IPMs. One of the seminal papers on such methods is by Wright [14], which shows a sub-quadratic complexity can be achieved given that the starting point is a positive infeasible solution and the problem has a strict complementarity solution. Comparisons have been made between feasible and infeasible interior point methods. In [15], Wright states that although feasible IPMs are theoretically faster than infeasible IPMs by a factor of n, where n is the size of the input, in practice, both methods can be highly effective for solving linear programs.

IPMs have been implemented with great success in recent years. In fact, major optimization software such as CPLEX and Gurobi provide the ability to use IPM to solve LPs. For the purposes of this paper, we use the feasible interior point method as described in [9] and discuss it in further detail in Subsection 4.1.

Quantum computing was first experimented with in the late 1990s, and by the early 21st century, the first scalable quantum computer was developed. The simulation of molecular and chemical systems was the first successful application of quantum computing, demonstrating its potential in solving complex problems. Quantum computing is also utilized in cryptography, optimization, and machine learning. Its promise to solve complex problems faster than the classical counterparts makes it an interesting technology for various industries, including finance, healthcare and manufacturing.

While quantum computers that outperform classical computers in solving equations do not currently exist, research into potential breakthroughs is deemed necessary to achieve quicker processing times. Over the years, many proposals for quantum computers that can solve linear systems of equations have been made. One of the earliest practical implementations was done by Barz et al. [16], achieving a solution vector fidelity of 64-98%. Cai et al. [17] later surpassed this achievement around the same time, with a fidelity ranging from 82.5% to 99.3%. Finally, in 2018, Wen et al. [18] demonstrated the first implementation for solving an 8x8 linear system of equations.

Quantum linear system algorithms (QLSAs) have been applied to optimization problems since the early 2020s. One such application is the development of quantum interior point methods (QIPMs), which various researchers have introduced. In 2020, Casares et al. [19] proposed a polylog algorithm for a feasible interior predictor-corrector algorithm using QLSAs. This algorithm corrects a prediction of the optimal solution iteratively and starts from an initial point in the feasible region. In 2022, Mohammadhossein et al. [20] investigated several QIPMs and proposed an infeasible interior point method that outperforms some feasible IPMs. More recently, in 2023, Zeguan et al. [21] proposed a feasible IPM with improved complexity bounds for solving ℓ_1 norm soft margin support vector machine problems. These developments demonstrate the potential of QLSAs in optimization and highlight the ongoing efforts to develop efficient QIPMs for practical applications.

3. Motivation and Contributions

Most IPMs employ iterative techniques that solve complex and time-consuming linear systems of equations as an intermediate stage. Solving such systems with numerous variables and constraints becomes particularly challenging. Consequently, a demand exists for more effective and precise approaches to address these problems. Our objective is to alleviate the computational burden

associated with solving linear equations in the classical domain by leveraging quantum algorithms. This is achieved through the use of the HHL algorithm.

Our contributions include,

- Implementation of a novel modification to the feasible Interior Point Method for solving LPs. The modification replaces the equation-solving step with the quantum HHL method.
- Provide extensive analysis of the modification to IPM using simulation in qiskit.
- To establish a baseline for comparison, we consider basic IPM, IPM which uses the full Newton system (with the non-linear terms) and a linearization using McCormick relaxations. The simulation results for the baseline using McCormick relaxation are not reported here. Please see the thesis by Adoni [22] for details on the baseline algorithms.

4. Preliminaries

In this section, we will discuss two important concepts that are referred to frequently in this paper. The first is the feasible IPM, as introduced in [9]. This is used to solve a maximization problem given below:

$$\begin{aligned} &\max\ c^T x \\ &\text{s.t.} \\ &Ax \leq b \\ &x \geq 0 \end{aligned} \tag{1}$$

In this problem, $A \in \mathbb{R}^{mxn}$, $x \in \mathbb{R}^n$ represents the primal variable, $c \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$.

Later on, we will provide a brief description of the HHL algorithm, which is used to solve a set of linear equations.

4.1. Feasible IPM

Feasible Interior Point Methods (IPMs) are techniques employed to solve linear optimization problems when an initial point and a path leading to the optimal solution exist within the solution space. In order to begin the process, the Lagrangian of the log-barrier problem (2) is analyzed, as explained in [9].

$$\begin{aligned} L(x,d,y) = c^T x + \mu(\sum_j \log x_j + \sum_i \log d_j) \\ + y^T(b - Ax - d) \end{aligned} \tag{2}$$

where, y is a vector of dual variables, $0 < \mu \leq \infty$, and d, q are a vector of slack variables. Critical points are points where the first order derivatives vanishes. The following equations are satisfied by critical points.

$$\begin{aligned} \frac{\partial L}{\partial x_j} &= c_j + \mu\frac{1}{x_j} - \sum_i y_i a_{ij} = 0 && \forall j = 1,2,...,n \\ \frac{\partial L}{\partial d_i} &= \mu\frac{1}{d_i} - y_i = 0 && \forall i = 1,2,...,m \\ \frac{\partial L}{\partial y_i} &= b_i - \sum_j a_{ij}x_j - d_i = 0 && \forall i = 1,2,...,m \end{aligned}$$

The equations above can further be simplified to,

$$Ax + d = b \tag{3}$$

$$A^T y - q = c \tag{4}$$

$$XQe = \mu e \tag{5}$$

$$YDe = \mu e \tag{6}$$

where we substitute $q = \mu X^{-1}e$ and X, Y are diagonal matrices given by vectors x, y respectively. In feasible IPM, we start with an arbitrary point (x, y, d, q) that is either primal or dual feasible. It is essential that each vector in the point (x, y, d, q) is non-negative. The next step involves determining the direction $(\Delta x, \Delta y, \Delta d, \Delta q)$ that the point must traverse to converge towards the μ-centre on the central path. This step brings the new point $(x + \Delta x, y + \Delta y, d + \Delta d, q + \Delta q)$ closer to the μ-center. To do this, we modify equations 3 - 6 by replacing x with Δx and the other variables as necessary. Finally, solve the resulting set of equations to obtain the direction.

$$\begin{bmatrix} A & 0 & I & 0 \\ 0 & A^T & 0 & -I \\ Z & 0 & 0 & X \\ 0 & W & Y & 0 \end{bmatrix} \times \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta w \\ \Delta z \end{bmatrix} = - \begin{bmatrix} Ax + w - b \\ A^T y - z - c \\ XZe - \mu e \\ YWe - \mu e \end{bmatrix} \tag{7}$$

where we substitute a variable, ρ in place of $b - Ax - w$, σ in place of $c - A^T y + z$, and γ in place of $z^T x + y^T w$.

Computing a solution to Equation 7 is known as the Newton step. The Newton step has a classical worst-case time complexity of $O(ns\kappa \log(\frac{1}{\epsilon}))$, where κ is the condition number of the input matrix and ϵ is the desired output precision and s is the sparsity of the matrix. Standard methods used to solve the linear set of equations require storing the entire matrix of coefficients in memory, along with intermediate results obtained during multiple passes through the matrix. This leads to a significant growth in memory requirement when using classical algorithms. A Newton step can be executed more efficiently using the HHL algorithm. This is provided we can load the right-hand side efficiently and the solution vector (or a near approximation) can be recovered efficiently. The first assumption is not limiting; however, the second assumption is strong and requires more progress.

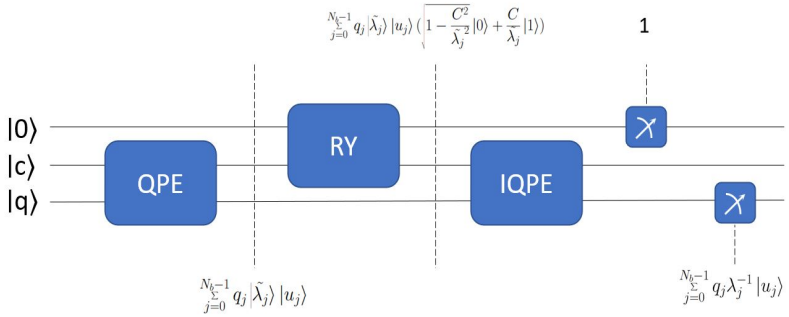

Figure 1: HHL Circuit Diagram

4.2. HHL

Given a linear system of equations $Gq = r$, we represent q, r by a super positions $|q\rangle, |r\rangle$. The solution is encoded in state $|q\rangle$ and is given by the following equation,

$$|q\rangle = G^{-1} |r\rangle$$

Writing the matrix G and vector r in the eigenbasis of G, the equation can be rewritten as,

$$|q\rangle = \sum_{i=0}^{N-1} \lambda_i^{-1} r_i |u_i\rangle \tag{8}$$

where N is the size of vector r and u_i denotes the i^{th} eigenvector of G.

To solve a linear system of equations, we require a circuit that can determine and invert the eigenvalues of matrix G. This circuit will produce a state that matches the solution to the system of equations. Figure 1 shows one circuit that can accomplish this.

To start, we run the quantum phase estimation (QPE) algorithm. This step is crucial in obtaining the eigenvalues of the matrix G and storing them in the auxiliary register. The resulting state will be in the eigenvector basis of G. The second stage of the HHL algorithm is the inversion stage. In classical algorithms, computing the inverse of a matrix is a time-consuming task, especially for matrices of large size. However, in quantum algorithms, matrix inversion can be performed much more efficiently using an RY gate conditioned on the eigenvalues.

$$\begin{bmatrix} \cos\psi/2 & -\sin\psi/2 \\ \sin\psi/2 & \cos\psi/2 \end{bmatrix}$$

where $\psi = 2 \arcsin \frac{C}{\lambda}$; for some constant C bounded by the smallest eigenvalue λ. A measurement of 1 in the auxiliary bit after applying the RY gate gives us the inverse eigenvalues.

$$RY_\theta(|0\rangle) = \sum_{j=0}^{N-1} r_j |\lambda_j\rangle |u_j\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle\right)$$

To complete the description of the HHL algorithm, the result, which is in the eigenvector basis, needs to be converted to the computational basis by applying the inverse Fourier transform. The inverse quantum phase estimation (IQPE) block shown in Figure 1 achieves this step. In case the measurement outcome is not equal to 1 after implementing the RY gate, the entire process should be repeated to ensure accurate results.

5. Quantum Accelerated Central Path Method

The series of steps in the central path method are outlined in Algorithm 1. We begin with a feasible interior point $(x, y, w, z) > 0$ and continue to solve the Newton step using the HHL algorithm until we reach an approximate optimal answer with a tolerance of ϵ. To ensure no point receives a negative value, we select a step size θ. It should be noted that we use different step sizes for primal variables θ_1 and dual variables θ_2 in our implementation, as this leads to faster convergence.

In order to use the HHL algorithm for solving a linear system of equations, it's important to provide the correct type of input. There are a few key things to keep in mind, such as:

Algorithm 1: Quantum Accelerated Central Path Algorithm

Input: $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$
Output: x
Set: $(x, y, w, z) > 0$
while $\gamma \geq \epsilon$ **do**
  Set: $\rho = b - Ax - w$
  Set: $\sigma = c - A^T y + z$
  Set: $\gamma = z^T x + y^T w$
  Set: $\mu = \delta \frac{\gamma}{n+m},\ 0 < \delta < 1$
  Compute step directions $(\Delta x, \Delta w, \Delta y, \Delta z)$ using HHL (8)
  Compute step size using
  $x = x + \theta_1 \Delta x$
  $w = w + \theta_1 \Delta w$
  $y = y + \theta_2 \Delta y$
  $z = z + \theta_2 \Delta z$

1. Matrix type: The input matrix should be Hermitian because it must have real eigenvalues to perform phase estimation accurately. This also allows for simulation of the Hamiltonian.
2. Matrix condition number: The condition number of the matrix is an important consideration. A well-conditioned matrix with a low condition number is easier to invert using the HHL algorithm. In contrast, a poorly conditioned matrix can result in more auxiliary qubits and longer runtimes.
3. Number of registers available: The HHL algorithm provides an approximate solution with a precision that depends on the number of auxiliary qubits and the accuracy of the phase estimation step.
4. Matrix sparsity: The sparsity of the matrix A with only a few non-zero entries. It also affects the number of gates required for the algorithm and the overall runtime.

We can control the type and the number of registers. However, the sparsity pattern and the condition number are not under our control, and affect the run time of HHL.

5.1. Complexity Analysis

To determine the runtime of the algorithm, we can calculate the number of iterations required by the IPM and the complexity of the HLL algorithm in each iteration, which is expressed below.

Theorem 5.1. *Consider the linear optimization problem defined by 1. The complexity to get an ϵ-solution, i.e., a solution that satisfies $z^T x + y^T w \leq \epsilon$, is*

$$O(N \frac{s^2 \kappa^2}{\epsilon} (\log N)^2)$$

where N is the size of the input vector, s is the maximum sparsity of the matrix in any Newton step. Similarly, κ is the largest condition number seen in any Newton step and ϵ is the desired accuracy.

Proof. The proof for this theorem stems from the fact that the bound for the number of iterations required by central path method is given by [23],

$$O(N \log \frac{N}{\epsilon})$$

The complexity of the HHL method [24] is

$$O(\log(N) \frac{\kappa^2 s^2}{\epsilon})$$

where κ, s are for the matrix in (3)–(6). Therefore the complexity is $O(N \frac{s^2 \kappa^2}{\epsilon} (\log N)^2)$. ■

If S is dense then we can use a version of HHL [25] with complexity $O(\sqrt{N}(\log N)\kappa^2/\epsilon)$ to obtain a bound as above.

5.2. Discussion

Before delving into the section on experiments, we will address some limitations and suggest modifications for a more practical implementation of the new central path approach. Here are some essential points to consider:

- QPE precision and computation time have a trade-off relationship. Increasing the precision may result in longer computation time and vice versa.
- The HHL algorithm has restrictions on input, so it is crucial to start with optimal instances to ensure favourable outcomes. This is especially important for practical applications.
- Generating a time-evolved matrix from Hermitian input is difficult in practical settings. So far, only local and sparse Hamiltonians have been identified as satisfying the necessary criteria.
- QPE has a range from 0 to 2^t. Input values outside of this range cannot be used for the QPE process. It is important to consider the range of values to effectively utilize the QPE method.

6. Measures of Performance

In the HHL method of Qiskit, we set the circuit's accuracy as $\epsilon = 10^{-3}$ in the constructor. Furthermore, we outline several metrics that will be examined in the outcomes.

6.1. State Fidelity

State fidelity is a way to measure how similar the actual state of a quantum system is to the desired target state. It helps us evaluate how well quantum operations and algorithms are performing. State fidelity values range from 0 to 1, with a score of 1 indicating an exact match between the actual and target states. If the score is less than 1, there is some deviation from the desired state. We can use the built-in `state_fidelity()` method in Qiskit to calculate this value.

6.2. SSE

We assess the effectiveness of the HHL approach using the Sum of Squared Estimate of Errors (SSE) metric. This measures the difference between the actual data and the estimation model. To calculate the SSE, we subtract the estimated data from the actual data, square the difference, and add up all the squared differences. The SSE serves as a numerical indicator of how accurately the HHL method solves the linear system of equations. The smaller the SSE, the higher the accuracy of the HHL method, and conversely.

$$SSE = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

where y_i is a given value and $f(x_i)$ is the result (output) of the Quantum Central Path Method.

We will establish naming conventions for the different types of Central Path Methods used. To ensure clarity, we will refer to the Central Path Method that utilizes Newton's method as the "Linear Central Path Method." The approach that uses the non-convex solver provided by Gurobi will be called the "Nonlinear Central Path Method." Lastly, we will refer to the Central Path Method that implements the quantum HHL algorithm as the "Quantum Central Path Method." This way, we can easily distinguish between the different methods being considered.

7. Experimental Setup

We rely on specialized software packages to ensure the effective implementation and operation of the Central Path Method. The Julia programming language is the foundation for our work due to its compatibility with other languages and systems, making it easy to integrate with tools such as Qiskit. Additionally, Julia has a growing ecosystem of packages dedicated to linear optimization and mathematical programming, including the popular JuMP library for modelling and solving optimization problems. There are several reasons why this library is essential for this work, including its user-friendly syntax, flexibility in changing solvers, and ability to communicate with most solvers in memory, eliminating the need for intermediary files. Benchmarking shows that JuMP can create problems at similar speeds to special-purpose modelling.

The JuMP library is an excellent tool that works well with different solvers. We'll be using the Gurobi solver because it has a wide range of parameters that allow you to control how it works. Two parameters that stand out as particularly useful are "Method" and "NonConvex." For example, by setting the "Method" parameter to 2, you can instruct the solver to use interior point methods when solving linear programming problems. Doing this makes comparing results obtained from the in-built optimizer with the Central Path Method easier. Moreover, setting the "NonConvex" parameter to 2 allows Gurobi to tackle nonlinear problems more efficiently by introducing McCormick relaxations.

To experiment with the HHL algorithm, we used the Qiskit 0.36.1 library in Python instead of creating our implementation in Julia. We integrate with the Qiskit package in our Julia code using the "PyCall" package. We can then use the HHL circuit as a black box for our experimentation.

Simulation of the Quantum Central Path Method can be computationally demanding. This is true for complex problems requiring large intermediate memory and intricate constraints. Clusters, on the other hand, are specifically designed to handle large and complex calculations. They have specialized hardware and software that enable highly parallel and efficient processing. In this paper, we utilize the Cedar supercomputer provided by Alliance Canada to tackle these challenges.

To maximize the resources and save time, we submit jobs on Cedar as job arrays using the SLURM workload manager. Each job is limited to a maximum of 7 days and has access to 500 GB of memory. Job execution occurs on one of the 96 available nodes in this memory category. This approach allows for effective parallel processing and management of tasks, as they share many similarities.

We summarize the CPU usage statistics in Table 1. The table shows that 8.08 core years were used, equivalent to continuously running computations on a single CPU core for about eight years. These experiments require a high level of computational demand. Therefore, utilizing a cluster is essential.

7.1. Instance Generation

In this study, we focus on tridiagonal matrices, which have non-zero elements only on the main, lower, and upper diagonals. These matrices have well-studied algorithms for various operations, such as solving linear equations, finding eigenvalues and eigenvectors, and inverting matrices. We refer to them as "efficient" instances

Resource	Total CPU Usage (in core years)	Projected CPU Usage (in core years)
cedar-compute	8.08	9.27

Table 1
Compute Canada Usage Statistics

because they allow for a more accurate simulation of the HHL algorithm in Qiskit.

Instances for the Linear and Nonlinear Central Path Methods are as follows:

- Efficient Instances: The input matrix is tridiagonal, the vector b and the slack variables are vectors of ones. We start with a matrix size of 2×2 and go up to a size of 25×25.

Instances for the Quantum Central Path Method take the following form,

- Efficient Instances: The input matrix is tridiagonal, the vector b and the slack variables are vectors of ones. We start with a matrix size of 2×2 and go up to a size of 5×5.

8. Results and Discussion

Our main goal is to evaluate the performance of the quantum-accelerated HHL algorithm. To do this, we will compare it with the classical method of solving the Newton step. For the baseline results for the non-linear Newton step and the use of McCormick relaxations see [22]. We will analyze each method regarding accuracy, efficiency, and scalability. We will conduct numerical experiments on different test instances to gain further insights into their strengths and limitations. Our study will provide valuable information for future research on developing new mathematical methods and quantum algorithms that involve solving linear systems of equations.

Table 2 displays the outcomes of the Linear Central Path Method for efficient instances. Column 2 displays the number of iterations completed. Columns 3,4 are for the objective function value and time as computed by the Gurobi's built-in IPM solver. column 5 shows the optimal, and the final column, column 6, shows the amount of time taken by the Linear Central Path Method as implemented by us. We modify this reference implementation and replace the equation solving step with the HHL algorithm.

The data shown in Table 2 reveals that the Linear Central Path Method needs more iterations as the instance size grows. This implies that solving larger and more complex instances may need more computational resources. Nevertheless, the optimal values obtained from IPM Optimal are identical to those of the optimal.

Table 3 displays the results of the nonlinear Newton step on efficient instances. The first column lists the instances, while the next three columns show the number of iterations, objective function value, and the time it takes for Gurobi's built-in interior point solver. The last two columns reveal the objective value and the time taken by our implementation of the nonlinear Newton step when Gurobi is used to solve the nonlinear program (with option `non-convex=2`), which is explained in detail in [22].

However, compared to the results obtained using the Linear central path method, this method's computation time and optimal solution are less favourable. For instance, for a matrix size of 12×12, the Nonlinear Central Path Method took almost 10 hours to find a solution of value 6, while the optimal solution is 4. The last row of the table shows that when solving a 14x14 matrix, a large number of feasible solutions were searched and over 500 GB of memory was consumed. To improve the performance of the Nonlinear Central Path Method, implementing stricter bounds could potentially address this issue.

The Quantum Central Path method was used to analyze efficient instances and the results are presented in Table 4. The data indicates that the SSE values are low and the gap values are small, demonstrating the reliability and precision of the method. It is important to note that the 5x5 instance reached its maximum runtime limit of 7 days, but the final iteration values were still recorded. However, the results for the efficient instances are promising. The simulation of HHL is computationally expensive due to the input matrix not being Hermitian and the size not being a power of 2. To address this issue, a reduction is used to make it Hermitian and of the right size, which increases the state space that needs to be explored by the simulation. Table 5 shows that the state space for a 5x5 input is 2^{64}.

In Table 5, we can find detailed information about the computing resources used by the Quantum Central Path Method. As the instance size increases, the input matrix size required for the HHL algorithm also increases. This is because the matrix needs to be changed into a Hermitian matrix. Our experiments have found that the largest matrix size used was 64×64, which required approximately 50GB of memory. The CPU Efficiency column shows how long it would take to run the job with the given resources. For instance, for the 5x5 instance, the job took about 37 days to run.

Instance	Iterations	Optimal Value	Optimal Time (s)	IPM Optimal	IPM Time (s)
2 × 2	7	1	3.13	1	2.451
3 × 3	7	1	3.133	1	2.459
4 × 4	7	2	3.013	2	2.443
5 × 5	7	2	3.01	2	2.419
6 × 6	8	2	3.075	2	2.473
7 × 7	9	3	3.019	3	2.457
8 × 8	8	3	3.039	3	2.424
9 × 9	9	3	2.995	3	2.46
10 × 10	9	4	3.044	4	2.441
11 × 11	9	4	3.229	4	2.487
12 × 12	9	4	3.056	4	2.446
13 × 13	10	5	3.084	5	2.394
14 × 14	9	5	2.957	5	2.403
15 × 15	10	5	3.087	5	2.427
16 × 16	10	6	3.038	6	2.449
17 × 17	9	6	3.091	6	2.407
18 × 18	10	6	2.95	6	2.303
19 × 19	10	7	2.886	7	2.27
20 × 20	10	7	3.396	7	2.47
21 × 21	10	7	3.128	7	2.445
22 × 22	10	8	3.151	8	2.454
23 × 23	10	8	3.259	8	2.453
24 × 24	10	8	3.114	8	2.413
25 × 25	11	9	3.125	9	2.459

Table 2
Linear Solutions to Efficient Instances

Instance	Iterations	Optimal Value	Optimal Time (s)	IPM Optimal	IPM Time (s)
2 × 2	7	1	3.13	1	0.818
3 × 3	9	1	3.133	2	0.886
4 × 4	9	2	3.013	2	8.602
5 × 5	9	2	3.01	3	1.378
6 × 6	15	2	3.075	3	34.641
7 × 7	10	3	3.019	4	8.669
8 × 8	21	3	3.039	4	653.515
9 × 9	10	3	2.995	5	83.122
10 × 10	29	4	3.044	5	5762.589
11 × 11	10	4	3.229	6	477.242
12 × 12	29	4	3.056	6	33905.434
13 × 13	10	5	3.084	7	8655.362
14 × 14	0	5	2.957	MEMLIMIT	

Table 3
Nonlinear Solutions to Efficient Instances

Instance	Duality Gap	Optimal Value	IPM Optimal	IPM Time (s)	Minimum Fidelity	SSE
2 × 2	-0.272872201	1	1.112556415	6212.561031	0.999697	0.531
3 × 3	-0.459883783	1	2.083048712	19043.03069	0.999017	1.081
4 × 4	-0.257149333	2	2.027472051	107388.935	0.99688	0.63
5 × 5	2.854052961	2	-2.758604754	TIMELIMIT	0.998971	0.384

Table 4
Quantum Solutions to Efficient Instances

Instance	HHL Input Size	Iterations	CPU Efficiency	Memory Used (GB)
2×2	16×16	4	22 H	3.96
3×3	32×32	6	2 D 3 H	8.46
4×4	32×32	18	8 D 8 H	23.11
5×5	64×64	9	36 D 21 H	53.86

Table 5
Quantum Metadata for Efficient Instances

Note that the 4x4 instance requires 18 iterations, but the number of iterations needed for the other four cases is similar to the best classical method. The simulation took a long time because of the large search space. To conduct a definitive study, more efficient simulation methods are necessary due to the enormous state space. According to the data in Table 5, simulation studies are currently not feasible even on 6x6 instances. Using a quantum computer to run the proposed algorithm is also currently not possible until more efficient methods or quantum memory that can read and write state vectors are developed. Let A be of size $N \times N$ with sparsity s. If b is encoded as $\sum_{i=0}^{N-1} b_i |i\rangle$ then $O(\log N)$ qubits are needed in the encoding. The HHL algorithm has complexity $O(\log N s^2 \kappa^2/\epsilon$ that depends on the sparseness s and the condition number κ [10]. So only for sparse matrices an exponential speedup is observed compared to the best classical algorithms which have complexity $O(N\kappa)$. If A is dense then a modification of HHL by Wossnig et al. [25] has complexity $O(\sqrt{N}\log N\kappa^2/\epsilon)$. This algorithm for dense matrices is faster by a quadratic factor. The speedup for dense matrices is not exponential. There is a related issue of determining the quantum state from the measurements using a process known as quantum tomography. This step is needed to extract the solution x from the state vector $|x\rangle$. The complete solution can be determined by measuring the complete set of observables whose expectations characterize the state [26]. For most quantum states the probabilities of observing some specific basis state can be very low (exponentially small in N). This means an exponential number of measurements (in N) are needed to determine x completely. For states that are matrix product states (MPS) efficient methods are known for estimating x [27]. The complexity of the proposed algorithms with methods using quantum tomography to recover x from $|x\rangle$ is $\Omega(N + \sqrt{N}\log N\kappa^2/\epsilon)$. This also explains the large running times in Table 5. No efficient methods are known for quantum tomography in general. However, there are several types of MPS states that be efficiently estimated from the quantum state and for such states the method proposed here can be efficient [27].

9. Conclusion

Our research proposes a new way to tackle linear programming problems by combining the HHL algorithm with the central path method. We examined three versions of the Central Path Method: the Linear Central Path Method, the Nonlinear Central Path Method, and the Quantum Central Path Method, and conducted extensive experimentation on tridiagonal instances. Though the HHL algorithm was impressive, the quantum accelerate central path algorithm simulation took too long and could not complete some instances in the allocated time. The issue of loading the right-hand side, solution extraction, and intermediate matrices' condition number needs to be further examined. However, integrating quantum algorithms with classical optimization methods can solve large-scale linear programs. Our study highlights the potential of quantum algorithms in solving optimization problems. It suggests that further optimization of the integration process and exploration of the practical applications of this approach in data science is necessary.

Acknowledgments

The authors thank Robert Benkoczi for valuable discussions. The authors thank the reviewers for suggestions that helped improve the presentation and highlight the relevance of this work to Data Science. The authors also thank David Neufeld for detailed comments on a draft which helped improve the readability.

The NSERC Discovery Grant provided support for this research. Additionally, the Digital Research Alliance of Canada (https://alliancecan.ca) played a part in enabling this research.

References

[1] J. Zhu, S. Rosset, R. Tibshirani, T. Hastie, 1-norm support vector machines, Advances in neural information processing systems 16 (2003).

[2] J. Yang, Y. Zhang, Alternating direction algorithms for ℓ_1-problems in compressive sensing, SIAM journal on scientific computing 33 (2011) 250–278.

[3] M. Yuan, High dimensional inverse covariance ma-

trix estimation via linear programming, The Journal of Machine Learning Research 11 (2010) 2261–2286.
[4] B. Recht, C. Re, J. Tropp, V. Bittorf, Factoring nonnegative matrices with linear programs, Advances in neural information processing systems 25 (2012).
[5] O. Meshi, A. Globerson, An alternating direction method for dual MAP LP relaxation, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II 22, Springer, 2011, pp. 470–483.
[6] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, I. Dhillon, Towards fast computation of certified robustness for relu networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 5276–5285.
[7] E. Wong, Z. Kolter, Provable defenses against adversarial examples via the convex outer adversarial polytope, in: International conference on machine learning, PMLR, 2018, pp. 5286–5295.
[8] G. Fung, R. Rosales, R. B. Rao, Feature selection and kernel design via linear programming., in: IJCAI, 2007, pp. 786–791.
[9] R. J. Vanderbei, et al., Linear programming, Springer, 2020.
[10] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Physical review letters 103 (2009) 150502.
[11] G. B. Dantzig, Maximization of a linear function of variables subject to linear inequalities, Activity analysis of production and allocation 13 (1951) 339–347.
[12] N. Karmarkar, A new polynomial-time algorithm for linear programming, in: Proceedings of the sixteenth annual ACM symposium on Theory of computing, 1984, pp. 302–311.
[13] C. Roos, T. Terlaky, J.-P. Vial, Interior point methods for linear optimization (2005).
[14] S. Wright, A path-following infeasible-interior-point algorithm for linear complementarity problems, Optimization Methods and Software 2 (1993) 79–106.
[15] S. J. Wright, Primal-dual interior-point methods, SIAM, 1997.
[16] S. Barz, I. Kassal, M. Ringbauer, Y. O. Lipp, B. Dakić, A. Aspuru-Guzik, P. Walther, A two-qubit photonic quantum processor and its application to solving systems of linear equations, Scientific reports 4 (2014) 6115.
[17] X.-D. Cai, C. Weedbrook, Z.-E. Su, M.-C. Chen, M. Gu, M.-J. Zhu, L. Li, N.-L. Liu, C.-Y. Lu, J.-W. Pan, Experimental quantum computing to solve systems of linear equations, Physical review letters 110 (2013) 230501.
[18] Y. Subaşı, R. D. Somma, D. Orsucci, Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing, Physical review letters 122 (2019) 060504.
[19] P. A. Casares, M. A. Martin-Delgado, A quantum interior-point predictor–corrector algorithm for linear programming, Journal of physics A: Mathematical and Theoretical 53 (2020) 445305.
[20] M. Mohammadisiahroudi, R. Fakhimi, T. Terlaky, Efficient use of quantum linear system algorithms in interior point methods for linear optimization, arXiv preprint arXiv:2205.01220 (2022).
[21] Z. Wu, M. Mohammadisiahroudi, B. Augustino, X. Yang, T. Terlaky, An inexact feasible quantum interior point method for linearly constrained quadratic optimization, arXiv preprint arXiv:2301.05357 (2023).
[22] V. Adoni, A quantum accelerated approach for the central path method in linear programming, Master's thesis, University of Lethbridge, Faculty of Arts and Science, 2023.
[23] J. Peng, C. Roos, T. Terlaky, Self-regular functions and new search directions for linear and semidefinite optimization, Mathematical Programming 93 (2002) 129–171.
[24] A. Abbas, S. Andersson, A. Asfaw, A. Corcoles, L. Bello, Y. Ben-Haim, M. Bozzo-Rey, S. Bravyi, N. Bronn, L. Capelluto, et al., Learn quantum computation using qiskit, chapter Investigating Quantum Hardware Using Quantum Circuits: Measurement Error Mitigation (2020).
[25] L. Wossnig, Z. Zhao, A. Prakash, Quantum linear system algorithm for dense matrices, Physical Review Letters 120 (2018).
[26] K. Vogel, H. Risken, Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase, Physical Review A 40 (1989) 2847.
[27] M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin, Y.-K. Liu, Efficient quantum state tomography, Nature communications 1 (2010) 149.