Gordon Cui

Quantum Algorithm for Exact Solutions in the Multi-Dimensional Knapsack Problem

Faculty of Environment and Information Studies
Keio University

Supervisor

Professor Rodney Van Meter

July 28, 2024

Abstract

The Multi-Dimensional Knapsack Problem (MDKP) is a strongly NP-Hard combinatorial optimization problem that extends the well-known Knapsack Problem (KP) from a single weight constraint problem to a multiple constraint problem. Despite the MDKP's numerous applications, ranging from finance to computer optimization, very few exact solutions have been studied for both classical and quantum computing. Classically, approaches for solving the MDKP often struggle with complexity as problem size increases due to the multi-objective nature of the problem. Thus, it is imperative we find more efficient solutions for tackling the MDKP.

Quantum computing offers a potential solution to solving such combinatorial optimization problems, with its plethora of quantum algorithms. In particular, Grover's search algorithm is a quantum unstructured database search that offers a quadratic speedup compared to its classical counterparts. This study will explore the application of Grover's search for the MDKP and demonstrate potential methods that can be used to solve the MDKP more efficiently.

By leveraging a combination of quantum search and classical pre and post-processing techniques, we propose an iterative quantum maximum-finding algorithm and circuit design that integrates classical adjustments to a quantum oracle, progressively refining solutions until the optimal solutions are found. Evaluations show that our method offers reduction in depth, width, optimization iterations, and runtime compared to approaches using preexisting search methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Background

With the advent of quantum computing, a multitude of quantum algorithms have emerged, demonstrating potential in solving complex problems across various fields [19, 20]. By leveraging the unique principles of superposition and entanglement of quantum computing, quantum algorithms can achieve significant speedups in hard problems that classical computing cannot. One notable example is Grover's search algorithm, an unstructured database search algorithm which offers a quadratic speedup compared to classical search algorithms [11]. Grover's algorithm demonstrates high potential in the field of combinatorial optimization, a domain where classical methods often face significant challenges. However, there are still many combinatorial optimization problems that remain relatively unexplored, with the Multi-Dimensional Knapsack problem being one of those problems.

1.2. Problem Definition

The Multi-Dimensional Knapsack Problem (MDKP) is a strongly NP-hard variant of the well-known Knapsack Problem (KP). While the traditional KP involves selecting a subset of items to maximize the total value without exceeding a single weight constraint, the MDKP extends this by introducing multiple con-

straints. Specifically, each item in the MDKP has multiple attributes (e.g. weight, volume, quality) and each attribute has a corresponding capacity limit. The goal is to select a subset of items such that the total value is maximized while satisfying all the given constraints.

Formally, consider a set of n items, each with m attributes. Let v_i be the value of item i, w_{ij} be the corresponding attribute of item i with respect to constraint j, and c_j be the capacity limit for constraint j. The problem can be stated as:

$$\text{Maximize} \quad \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_{ij} x_i \leq c_j, \quad j = 1, 2, \ldots, m$$

$$x_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n$$

Here, x_i is a binary string indicating whether item i is included in the knapsack (1) or not (0). The MDKP is particularly challenging due to its combinatorial nature and the exponential growth of possible solutions with increasing n and m.

The motivation behind finding efficient solutions to the MDKP lies in its numerous practical applications in fields such as finance, resource allocation, and logistics, where multiple constraints must be considered simultaneously. Traditional classical approaches often struggle with the complexity of the MDKP, especially as the problem size increases. There currently exists no efficient exact classical solution to the MDKP, with brute force exhibiting a complexity of $O(2^n \cdot nm)$. This demonstrates the need for more efficient and scalable solutions, such as those potentially offered by quantum computing.

1.3. Contribution

We propose a hybrid quantum-classical algorithm that uses novel pre-processing techniques, iterative quantum max-finding, and an oracle circuit design for finding exact solutions of the MDKP. To evaluate the effectiveness of our approach, we compare our algorithm utilizing our proposed methods against variants that employ preexisting methods. Evaluations using the IBMQ QASM simulator [12]

show the advantage of our method as it demonstrated superior efficiency in depth, width, elapsed cycles, and runtime. Furthermore, the reduction in growth of elapsed cycles and runtime in our proposed method show its potential scalability as problem item and attribute size grows.

1.4. Thesis Structure

Following the Abstract and Introduction sections, the rest of the thesis is structured as such: Chapter 2 contains background knowledge of quantum computing. Chapter 3 demonstrates the proposed method with the necessary information on the quantum algorithms and methods used. An example implementation and results of the experiment simulated with the IBMQ QASM simulator is presented in Chapter 4. Finally, chapter 5 concludes this thesis with reflection on acquired results and future methodology.

Chapter 2

Introduction to Quantum Information Theory

This chapter will introduce the basics of quantum information as a background for understanding the rest of the thesis.

2.1. Quantum Information

2.1.1 Qubits

A quantum bit, or qubit for short, is the basic unit of quantum computing, analogous to classical bits in classical information theory. Qubits are represented as a vector in a two-dimensional complex Hilbert space, typically denoted using Bra-Ket notation [15] to describe the state of qubit. A qubit can exist in one of two basis states, described as state vector $|0\rangle$ and $|1\rangle$,

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{2.1}$$

When representing multiple bits, a tensor product of qubit states are used. A two qubit state $|01\rangle$ can be represented as a tensor product of $|0\rangle \otimes |1\rangle$,

$$|0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \tag{2.2}$$

2.1.2 Superposition and Measurement

Contrary to classical bits, qubits are not limited to the two basis states of $|0\rangle$ or $|1\rangle$. A qubit can exist in a combination of quantum states, formally known as superposition. Qubits in superposition may be in states $|0\rangle$ and $|1\rangle$ at the same time, with different probabilities of measuring each state. The general state of a qubit, $|\psi\rangle$, is expressed as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.3}$$

Here, α and β are the probability amplitudes. When normalized, the probability of measuring the qubit in states $|0\rangle$ and $|1\rangle$ are given by $|\alpha|^2$ and $|\beta|^2$ respectively, where:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.4}$$

Measurement in quantum computing is the process of extracting classical information from quantum systems. By observing a qubit's state, we cause it to collapse from a superposition into one of the basis states $|0\rangle$ or $|1\rangle$. The act of measurement is probabilistic and results in the qubit taking on a definite state based on the probability amplitudes. Notably, we see that qubits exist as a fixed probabilistic distribution rather than a fixed value in the way classical bits do.

2.1.3 Bloch Sphere Representation

To visualize the state of a single qubit, the Bloch sphere provides a 3-Dimensional sphere representation where pure states correspond to points on the surface of the sphere, while mixed states are within the sphere. The north pole represents $|0\rangle$, the south pole represents $|1\rangle$, and the equator represents equal superposition states.

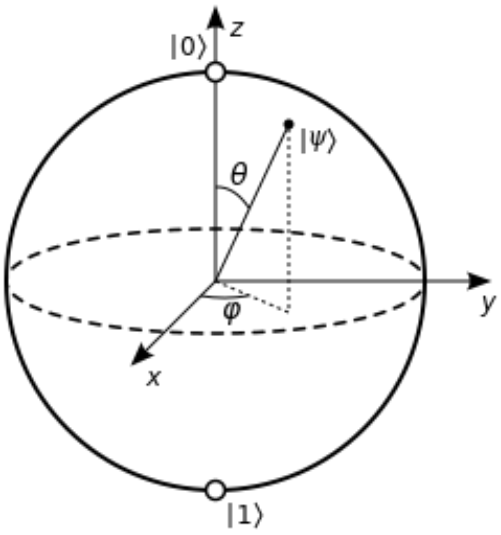


Figure 2.1: Bloch Sphere representing state ψ

2.2. Special Properties of Qubits

Qubits possess several additional unique properties particular to quantum physics. These properties include entanglement, decoherence, and the no-cloning theorem [22].

2.2.1 Entanglement

Entanglement is a quantum phenomenon where two or more qubits become interconnected such that the state of one qubit cannot be described independently of the state of the other. When qubits are entangled, the measurement of one qubit instantaneously affects the state of the other, regardless of the distance separating them. This property is central to many quantum algorithms and protocols, including quantum teleportation [9] and quantum cryptography [10].

Suppose we have quantum state:

$$\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{2.5}$$

This state is an example of an entangled state, where the qubits are not independent of each other. In this entangled state, we cannot observe the states $|01\rangle$

and $|10\rangle$ but instead, we can only observe the states $|00\rangle$ or $|11\rangle$. Entanglement implies that the states of the qubits are correlated. Specifically, for the state $|\psi\rangle$, the measurement outcomes of the two qubits are perfectly correlated:

- If the first qubit is measured and found to be in state $|0\rangle$, the second qubit will also be in state $|0\rangle$.
- If the first qubit is measured and found to be in state $|1\rangle$, the second qubit will also be in state $|1\rangle$.

2.2.2 Decoherence

Decoherence is the process by which a qubit loses its quantum properties and behaves more classically due to interactions with its environment. This loss of coherence results in the qubit no longer maintaining a well-defined quantum state, which can significantly affect the performance and accuracy of quantum computations.

Decoherence is a major challenge in building practical quantum computers because it can introduce errors into quantum calculations. Techniques such as quantum error correction are being developed to mitigate the effects of decoherence [6].

2.2.3 No-Cloning Theorem

The no-cloning theorem states that it is impossible to create an exact copy of an arbitrary unknown quantum state. Mathematically, if we have an unknown quantum state $|\psi\rangle$ and an attempt is made to create a copy $|\psi\rangle \rightarrow |\psi\rangle \otimes |\psi\rangle$, the no-cloning theorem proves that such a process cannot be achieved with a unitary operation. This problem proves challenging for error correction in quantum computing as classically, systems protect data from errors by making copies of data.

2.3. Operations on Quantum States

2.3.1 Quantum Gates

Quantum gates are unitary operators that manipulate and transform qubit states. These gates are reversible, meaning that the operations performed on qubits can be undone. When a quantum gate operation is performed on a qubit, it multiplies the qubit's state vector by the gate's matrix. Formally, we can represent a quantum gate operator as unitary operator U applied to quantum state $|\psi\rangle$, where U is a complex square matrix that when multiplied by its conjugate transpose $U^\dagger$, results in the identity matrix I. Mathematically, this is expressed as:

$$UU^\dagger = U^\dagger U = I.$$

A unitary gate can be any special unitary group of degree 2, SU(2), matrix, which can be defined as:

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \tag{2.6}$$

where $0 \leq \theta, \phi, \lambda \leq 2\pi$. Specific qubit gates can be constructed by substituting the appropriate angles. The most commonly used gates are named and they are shown in the following sections.

2.3.2 Single Qubit Gates

Hadamard Gate [H]

Creates superposition states by rotating the state 180 degrees around the XZ axis.

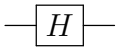


Figure 2.2: H Gate

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.7}$$

Pauli Gates [X] [Y] [Z]

Rotates state around their respective X, Y, Z axis. Specifically, X-gate flips the qubit state while the Y and Z gate introduces a phase shift.

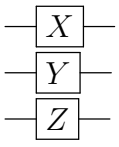


Figure 2.3: X, Y, Z Gates

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.8}$$

Rotation Operator Gates [Rx] [Ry] [Rz]

Rotates an arbitrary angle around their respective X, Y, and Z axis

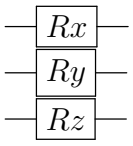


Figure 2.4: Rx, Ry, Rz Gates

$$R_x(\theta) = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$
$$R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{2.9}$$
$$R_z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

2.3.3 Multi-Qubit Gates

Controlled-NOT Gate [CX]

The CX gate is a two-qubit controlled-not gate that performs the X gate operation on the target qubit if the control qubit is $|1\rangle$. It does nothing if the control qubit is $|0\rangle$. The target gate is represented by a black dot and the control is represented by a plus sign enclosed in a circle.

Figure 2.5: CX Gate

$$\text{CX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.10}$$

Toffoli Gate [CCX]

The CCX gate is a three-qubit controlled-not gate with 2 control gates and 1 target gates. It performs the X gate operation on the target qubit if all control qubits are $|1\rangle$. If not, it does nothing.

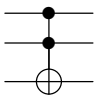


Figure 2.6: CCX Gate

$$\text{CCX} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.11}$$

2.4. Quantum Circuits

Quantum circuits are the fundamental framework for current quantum computation. These circuits are composed of multi qubit registers, a sequence of quantum gates, and measurements applied to a set of qubits, analogous to classical circuits that operate on bits. Quantum circuits are visualized using circuit diagrams, where time progresses from left to right. Each horizontal line represents a qubit, and quantum gates are applied sequentially along these lines. Figure 2.7 shows an example of a simple quantum teleportation circuit [9].

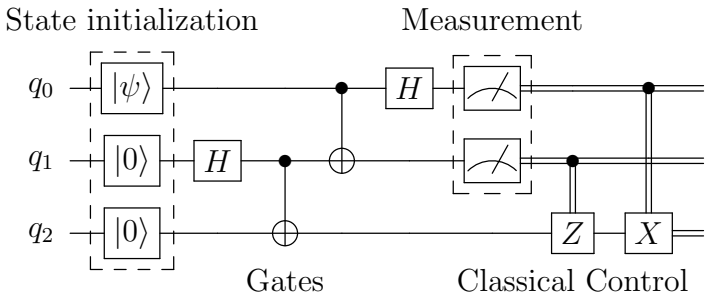


Figure 2.7: Quantum Teleportation Circuit

2.5. Quantum Fourier Transform (QFT)

The quantum fourier transform (QFT) is the quantum implementation of the classical discrete Fourier transform over the amplitudes of a wavefunction. Essentially, the QFT allows us to convert binary numbers to phases, and vice versa. Formally, the QFT maps state $|j\rangle$ as defined by:

$$|j\rangle \rightarrow \quad \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} |k\rangle \quad \text{where} \quad \omega = e^{2\pi i/N} \tag{2.12}$$

Intuitively, the QFT is a function transforms states between the Z computational basis and the Fourier basis. All multi-qubit states in the Z basis have corresponding states in the Fourier basis, and the QFT converts between the two bases. The ability to switch between bases is useful because we can perform phase operations in the Fourier basis that we could have not in the computational basis. One such use is arithmetic in the Fourier basis. For this we need to understand how numbers are represented in the Fourier basis.

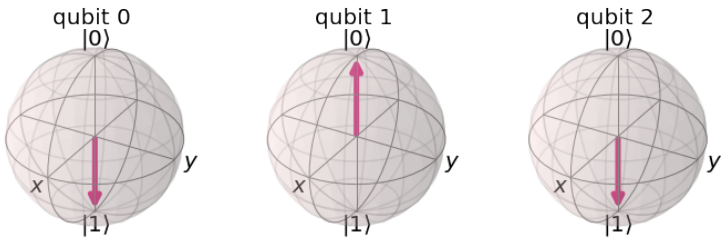


Figure 2.8: Bloch Sphere representing 5 in the computational basis

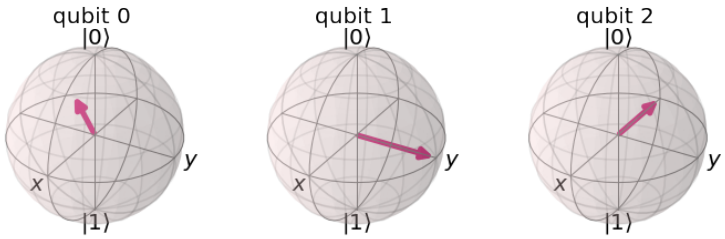


Figure 2.9: Bloch Sphere representing 5 in the Fourier basis

In the Z basis, numbers are stored in binary using states $|0\rangle$ and $|1\rangle$ with no phase rotations as shown in figure 2.8. However, in the Fourier basis, numbers are stored using different rotations around the Z-axis as shown in figure 2.9.

Chapter 3

Proposed methodology for solving the MDKP

3.1. Proposed Method

Figure 3.1 shows the outline of our proposed method using a combination of classical and quantum techniques. Our method consists of two classical pre-processing techniques and a iterative quantum maximum finding cycle that uses Grover's search algorithm to find optimal solutions.

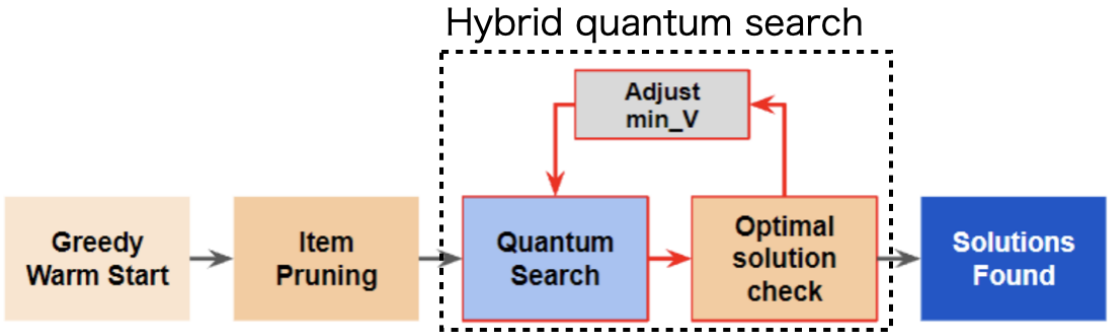


Figure 3.1: Proposed hybrid quantum MDKP algorithm process

3.1.1 Quantum maximum finding and Classical Pre-processing

We use a iterative max-finding loop with Grover's algorithm to find exact solutions. Each cycle, we seek solutions that meet attribute capacity constraints *cap* and a minimum value threshold min_v. If the solution found is not optimal, we adjust min_v and repeat until optimal solution(s) are found. Let v be a list of our item's values, w a 2d array representing the items' attributes across m attributes. Define $V_{\text{total}} = \sum_{j \in S} v[j]$ and $W_{\text{total}}[i] = \sum_{j \in S} w[i][j]$ for chosen set S.

However, an important factor in the efficiency of our optimization process is determining what initial minimum value threshold to start with. An initial threshold that is too low may cause excess iterations while a threshold that is too high may not have any possible solutions. Furthermore, there may be items that are highly sub-optimal and extremely unlikely to be in optimal solutions, thus wasting resources for consideration. To combat these two issues and improve efficiency and cost in the max-find, we employ the use of two classical pre-processing techniques:

Greedy Warm Start - A Greedy MDKP algorithm is used to efficiently find an initial minimum value, v_{min} by sorting items based on a value-to-attribute ratio. We select items from best to worst ratio until the capacity of knapsack is reached.

Item Pruning - Our item pruning method prunes items with bad ratios below the 30th percentile, as determined by the Greedy algorithm. Pruned items are removed as possible selected states during quantum search, reducing both search space and gate count.

Our max-finding algorithm proceeds as shown in Algorithm 1.

Algorithm 1: QMDKP (v, w, cap)

Input: v - list of item values
Input: w - list of item weights (each item can have multiple attributes)
Input: cap - list of capacity constraints for each attribute
Output: Optimal set of items meeting the constraints

```
V_total = Run Greedy Algorithm with input v, w, cap to get an initial
  solution and compute its total value;
min_v = V_total;
Run Item Pruning on v, w, cap to update the list v of item values;
while True do
    Grover's Search for optimal solutions with V_total > min_v and
      W_total[0..m] ≤ cap[0..m] for all indexes;
    if solutions found then
        Choose a solution uniformly at random and adjust min_v to the
          solution's V_total;
    else if no solutions found then
        return solutions
end
```

3.2. Grover's Search Algorithm

Grover's algorithm amplifies the amplitude of the desired state. In this study, the desired state corresponds to the combination of items that satisfies the current min_v and cap constraints. The quantum state evolved by Grover algorithm is defined by

$$|\psi(r)\rangle = (\hat{D}\hat{O})^r |\psi(0)\rangle . \tag{3.1}$$

The initial state is prepared in a uniform superposition given by

$$|\psi(0)\rangle = \frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^N} |\mathbf{x}\rangle , \tag{3.2}$$

where $\mathbf{x}$ is a binary encoding and N is total number of possible states. r is the number of Grover iterations. $\hat{O}$ is the oracle operator that distinguishes between solution and non-solutions by flipping the phase of solutions states by -1 while

keeping non-solution states the same:

$$\hat{O}\left|x\right\rangle = (-1)^{f(x)}\left|x\right\rangle. \tag{3.3}$$

$f(x)$ is oracle function and if $x = T$, $f(x) = 1$ and $f(x) = 0$ otherwise. $\hat{D}$ is the Grover diffusion operator given by

$$\hat{D} = 2\left|\psi(0)\right\rangle\left\langle\psi(0)\right| - \hat{I}. \tag{3.4}$$

The diffusion operator provides inversion about the mean to amplify the amplitude of marked states. We repeat this two operators for Grover iterations until $r = r_{\text{opt}}$. Traditionally, the number of Grover iterations for multiple solutions can be represented as such

$$r_{\text{opt}} = \frac{\pi}{4}\sqrt{\frac{N}{M}} \tag{3.5}$$

where N is number of possible states and M is the number of solutions. However, in an optimization problem like the MDKP, we do not know the number of desired solutions thus we employ the exponential search algorithm of Boyer [3] explained in the following subsection (3.2.1).

3.2.1 Finding Optimal Grover Iterations

To find the optimal number of Grover iterations, we call the exponential search of [3] as a subroutine that dynamically adjusts the number of iterations based on feedback from the search results. The exponential search function operates as follows:

1. Initialize $m = 1$ and set $\lambda = \frac{6}{5}$. (Any value of λ strictly between 1 and $\frac{4}{3}$ would do.)

2. Choose uniformly at random among the non-negative integers smaller than m.

3. Apply j iterations of Grover's algorithm starting from initial state $\left|\Psi_0\right\rangle = \frac{1}{\sqrt{N}}\sum_i \left|i\right\rangle$.

4. Observe the register.

5. If solutions exist, j is optimal number of Grover iterations, exit.

6. Otherwise, set m to $\min(\lambda m, \sqrt{N})$ and go back to step 2.

3.3. Oracle Circuit Design

3.3.1 Encoding and State Preparation

For item list v, we encode the selected items as a binary string. Specifically, 1 indicates that the items is selected while 0 represents that the item is not selected.

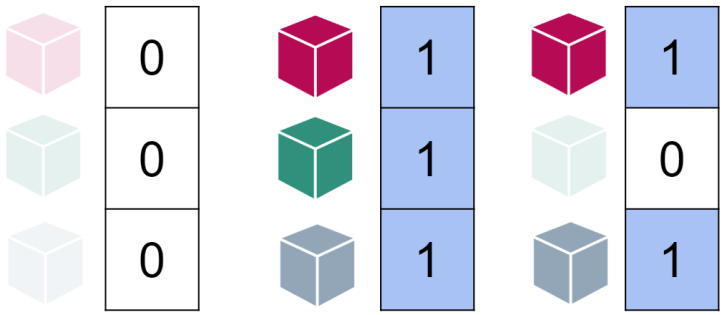


Figure 3.2: Example of encoding for n = 3 items, [0,0,0], [1,1,1], [1,0,1]

To represent this in our circuit, n items are encoded as n item qubits, with the order corresponding to the indices of v. We prepare them in an equal superposition state of $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ with Hadamard gates, with states $|0\rangle$ and $|1\rangle$ representing a non-chosen and chosen item respectively.

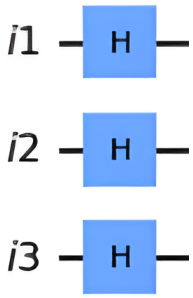


Figure 3.3: State preparation of n = 3 items

3.3.2 Circuit Construction

Our circuit is designed in two distinct parts: a value check and a capacity constraint check. Separating these checks helps to minimize the circuit width (qubit count). Our value check determines if the sum of the selected item values exceeds a given threshold. Our capacity constraint check ensures that all attributes of the selected items do not exceed their respective capacity constraints. Due to the multiple constraints of the MDKP, all constraint checks are executed in parallel to reduce circuit depth. If both value and constraint checks are true, our state is marked as a correct solution.

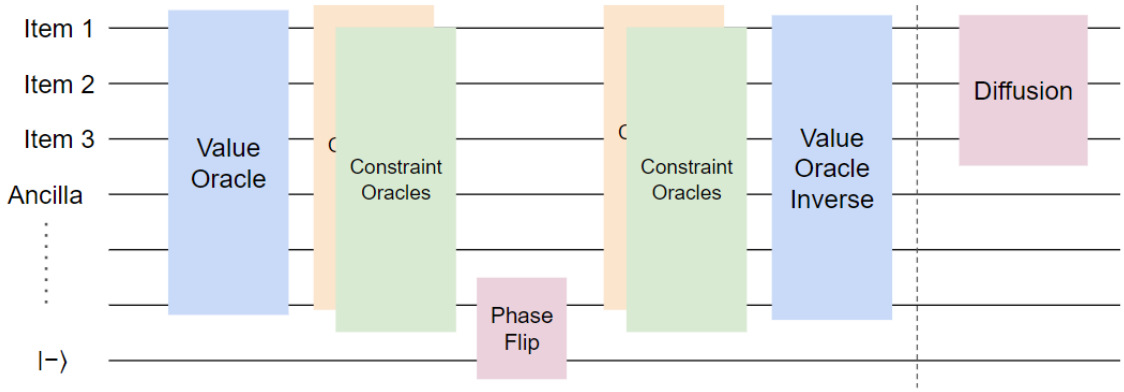


Figure 3.4: Value + Parallel Constraint Oracle

At the gate level, both checks operate essentially in the same way. For each constraint, we allocate k ancillary qubits to hold the respective sums. The size of k is chosen based on the binary length required to represent the maximum possible sum. Each item's respective values are added by our controlled direct QFT adder (discussed more in the following subsection), with the control being the corresponding item qubit. As the item qubits are in superposition, this controlled operation entagles the two. Essentially, the item qubits influence the adder, enabling the adder only if the item is "chosen" in the current state.

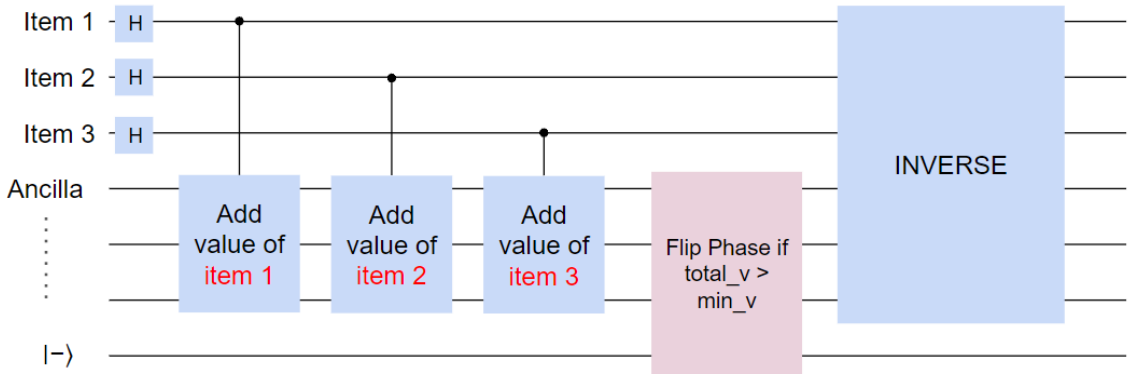
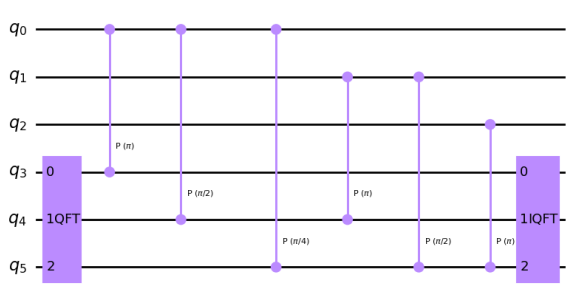


Figure 3.5: Value Oracle in depth

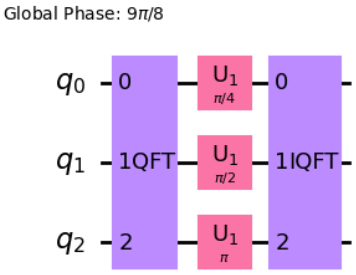
Direct QFT Adders

To perform the additions, the use of QFT is essential for the efficient implementation of quantum adders [5, 18, 21]. We use controlled direct QFT adders, our variant of Draper's QFT adder [7], to directly apply quantum addition from classical integer input.

Draper's original proposal was designed to add two integers, each encoded in separate quantum registers. The result of the addition is stored in one of these registers, which was also designated as the sum register. QFT is applied to the sum register, entering the Fourier basis, with a series of controlled rotations performed with control qubits from the second register. Finally, the inverse QFT is applied to the sum register, bringing it back to the computational basis with the sum of the two integers encoded. Figure 3.6 (left) displays Draper's original QFT adder.



(a) Draper QFT adder for 3 register addition



(b) Direct QFT adder for 3 register addition

Figure 3.6: Draper QFT adder vs our Direct QFT adder. Notice that our adder applies phase rotations directly, eliminating the need for ancillary qubits and additional gates.

However, as all values and attributes are known in advance, we can simplify the process by removing the control qubit register. Figure 3.6 (right) displays our version of a direct QFT adder. Instead of using a second register to hold one addition input, we directly implement all necessary rotations on qubits, using fewer gates and reducing qubit cost. Using the list of values and attributes as our inputs, we add them into their respective addition registers. Since all additions are consecutive, we only need to apply the QFT and inverse QFT gate for each register as seen in Figure 3.7.

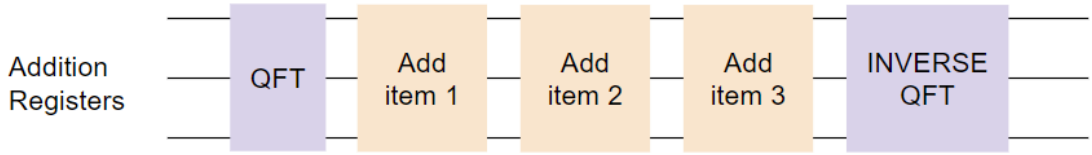


Figure 3.7: Consecutive QFT adders

Integer Comparator

To check if the current state satisfies value threshold requirements and attribute capacity constraints, comparator circuits are used to compare to specified classical integers. These classical integers are dynamically adjusted in our HQS cycle based off outputted solutions. Formally, this operator compares basis states

$|i\rangle_n$ against a classically given integer L of fixed value and flips a target qubit if $i \geq L$,

$$|i\rangle_n|0\rangle \mapsto |i\rangle_n|i \geq L\rangle. \tag{3.6}$$

Here, we use the Integer Comparator from the Qiskit library. It takes the input from our addition registers and compares it to a classical integer. This operation relies on the two's complement implementation of binary subtraction, using carry bits to determine the result. Specifically, the comparator tracks the most significant carry bit during the subtraction to decide if the input register value is greater than or equal to the classical integer. If the most significant carry bit is 1, the comparison is true, otherwise, it is false.

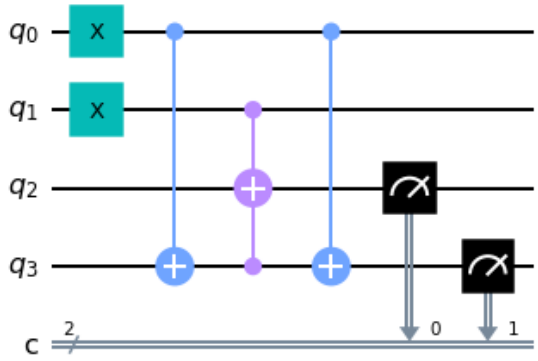


Figure 3.8: Comparing if qubit register is greater or equal to integer 3

Final Circuit

Below shows the entire main-oracle circuit (pre-inverse and diffusion) for a MDKP instance of n = 3 items and m = 2 attributes.

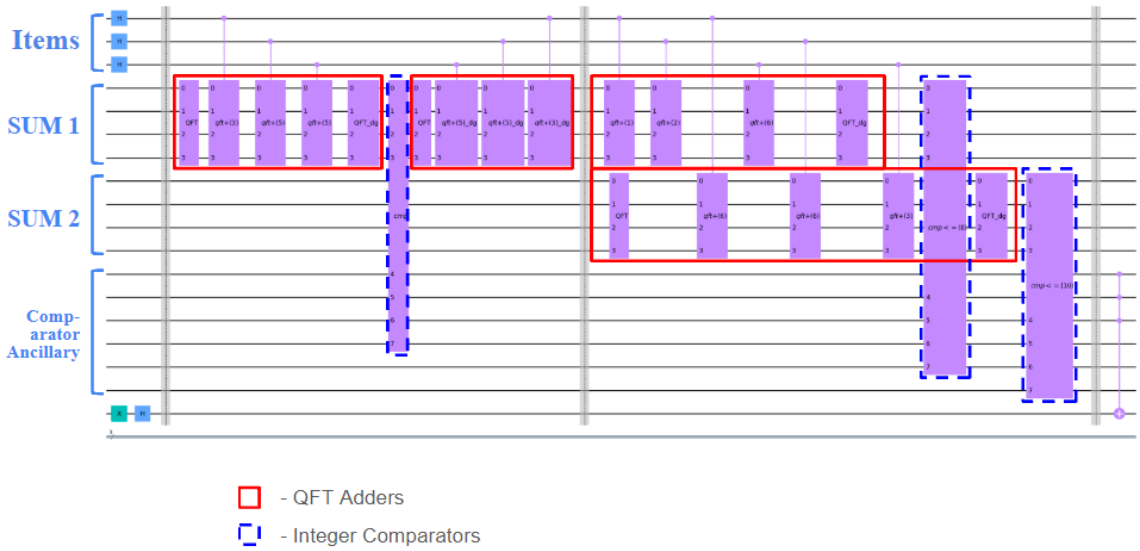


Figure 3.9: Circuit of our MDKP Oracle for 3 items and 2 attributes

Chapter 4

Implementation and Evaluation

4.1. Implementation

Here, we show an example implementation of the HQS MDKP algorithm running for problem size of n = 4, m = 2 with the following values, attributes, and knapsack capacities:

	Item 1	**Item 2**	**Item 3**	**Item 4**
Values	4	1	3	6
Attribute 1	1	4	1	1
Attribute 2	1	6	2	6

Capacities
/
5
7

Table 4.1: Values, Attributes, and Capacities

First we apply *MDKP Greedy* to find a good initial value threshold for our HQS. Greedy returns an output of:

```
Selected items: [0, 2] # Indices of Item 1 and 3
Total value of items: 7
```

Based off item value-attribute ratios, Greedy selects item 1 and 3, giving us a value of 7 to use as our initial minimum value threshold. Next we run our values through *Item Pruning*, pruning bad item ratios under the 25th percentile:

```
Prune item index:  1
```

Here, Item 2 is pruned and removed from consideration in our HQS oracle. Finally, we run our HQS process with the corresponding circuit for 10^4 shots and obtain the following output.

```
--- Problem: 0 , n = 4 , m = 2 ---
start loop 1 - min_v = 7
solution: 1001 totalv: 10 # More optimal solutions found!

start loop 2 - min_v = 10
END NEXT # No optimal solutions for min_v of 10

start loop 3 - min_v = 9
solution: 1001 totalv: 10 # Most optimal solution found
END
```

After measurement, we obtain the following results:

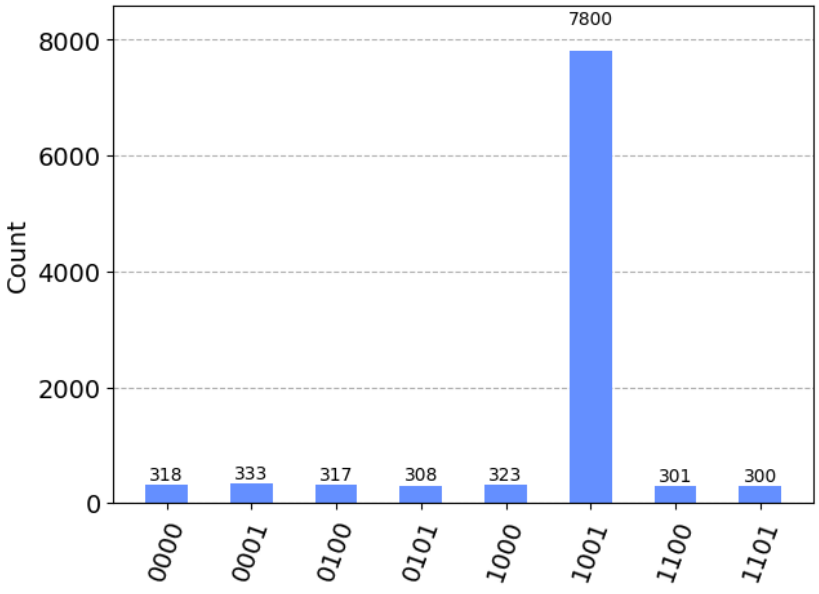


Figure 4.1: Measured probability distribution of n = 4, m = 2

Binary string of 1001 represents items 1 and 4 being chosen, giving us the correct optimal solution with a total value of 10 while staying below knapsack

capacity. We achieve a reasonable success probability of measuring our desired state of 78%.

4.2. Experimental setup

In order to evaluate the performance and scalability of our method, we must test for multiple instances with growth in both n items and m attributes. Our circuit was tested using the IBMQ QASM simulator [12] under two sets of experiments:

- **Experiment 1: Increasing n items with Constant m**
  - 20 problems per n
  - Each circuit run for 10^4 shots
- **Experiment 2: Increasing m with attributes Constant n**
  - 20 problems per m
  - Each circuit run for 10^4 shots

Due to the register size and gate count correlating with the binary representation size of our values and attributes, the performance of our circuit can vary even for problems of identical size. To ensure a fair testing environment, we generate 20 random instances for each problem size and compute the average performance required for a reasonable success probability of 50% upwards to approaching 100%.

In each experiment, we evaluate and compare three methods:

- **Method 1 - HQS(G + P):** Our hybrid quantum search with Greedy and Item Pruning (shown in Fig. 3.1) HQS
- **Method 2 - HQS(G):** Our hybrid quantum search with only Greedy
- **Method 3 - QMS:** The Quantum max search algorithm of [8]

As benchmarks, we take the averages of four parameters: elapsed cycles, runtime (s), circuit width, and circuit depth.

4.3. Results

4.3.1 Experiment 1: scaling with n

Table 4.2 and subsequent figures 4.2, 4.3, 4.4, and 4.5 show the results of the first set of experiments where problem size scales with n items.

n	m	HQS (G + P)		HQS (G)		QMS	
		Width	Depth	Width	Depth	Width	Depth
3	2	17.85	303.5	17.85	303.5	17.85	303.5
4	2	19.15	311.5	19.9	351	19.9	351.1
5	2	21.2	355.9	22.25	403.4	22.25	403.4
6	2	23.4	406.2	24	446.2	24	446.2
7	2	24.85	424.5	25	476.5	25	476.5
8	2	26	450.9	26.45	513.9	26.45	513.9

Table 4.2: Average required Width and Depth as n increases.

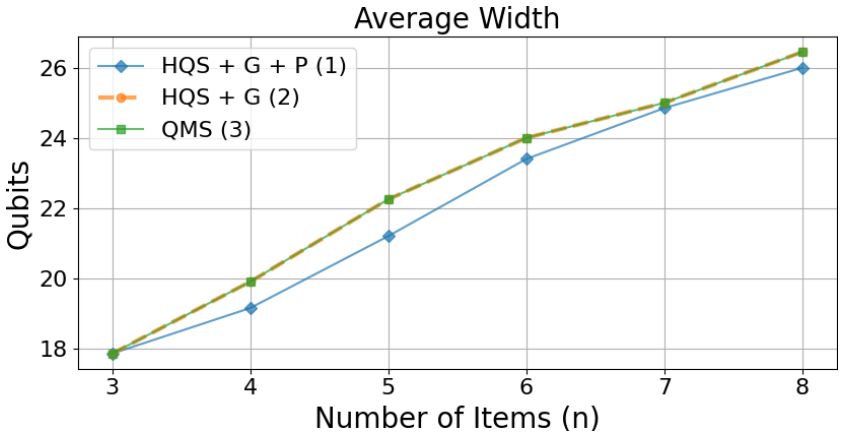


Figure 4.2: Average Width as n increases

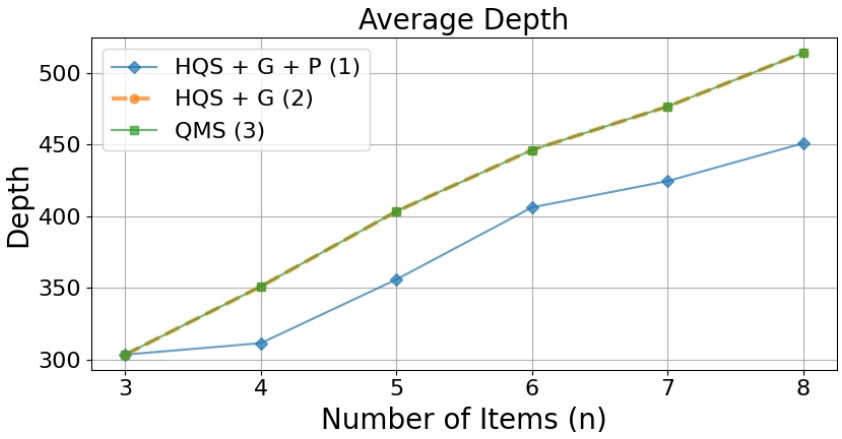


Figure 4.3: Average Depth as n increases

As n number of items increases, proposed method (1) shows reduced width and depth compared to methods (2) and (3). The difference in depth here is particularly noticeable, especially as the number of items increase. Methods (2) and (3) follow a similar trend, showing higher width and depth. These findings demonstrate the cost reduction benefits of incorporating our item pruning technique into quantum search algorithms.

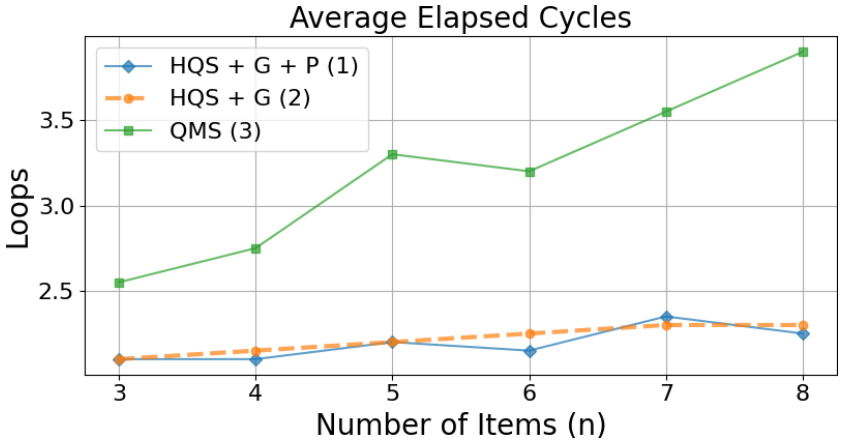


Figure 4.4: Average Elapsed Cycles as n increases

Examining the average elapsed cycles required to find optimal solutions, the quantum maximum search algorithm of (3) takes significantly more iterations compared to (1) and (2). Not only does method (3) start with a higher cycle

count, but it also exhibits a steeper increase in elapsed cycles as the problem size grows.

When comparing methods (1) and (2), the results indicate that on average, method (1) outperforms method (2), only losing in the case of $n = 7$. This small discrepancy may be attributed to outliers skewing the results, due to the relatively small sample size of 20 problem instances per size of n. Testing this with a larger number of problem instances may demonstrate an even more definitive advantage of method (1). Even so, we can already infer the potential scalability advantages of our proposed method in terms of elapsed cycles.

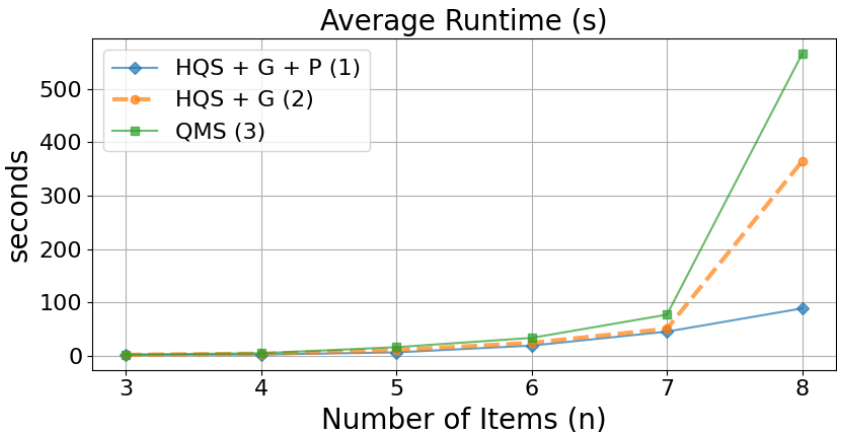


Figure 4.5: Average required Width and Depth as n increases.

The simulation runtime results show the scalability of our proposed method (1) as number of items increases. From n of 3 to 7, method (1) maintains lowest runtime followed by (2) and (3). As we approach n of 7 and 8, the performance gap widens significantly with method (3)'s runtime increasing sharply. Method (2) also experiences a noticeable increase in runtime, although still remaining more efficient than method (3). Conversely, method (1) shows superior scalability, maintaining a substantially lower runtime compared to the other methods. This trend clearly demonstrates the potential scalability of our method as problem size increases with n.

4.3.2 Experiment 2: scaling with m

Table 4.3 and subsequent figures 4.6, 4.7, 4.8, and 4.9 show the results of the second set of experiments where problem size scales with m number of attributes.

n	m	HQS (G + P)		HQS (G)		QMS	
		Width	Depth	Width	Depth	Width	Depth
4	1	13.8	254.5	14.2	289.9	14.2	289.9
4	2	18.85	308.1	20.05	353.6	20.05	353.6
4	3	24	320.3	25.2	367.7	25.2	367.7

Table 4.3: Average required Width and Depth as m increases.

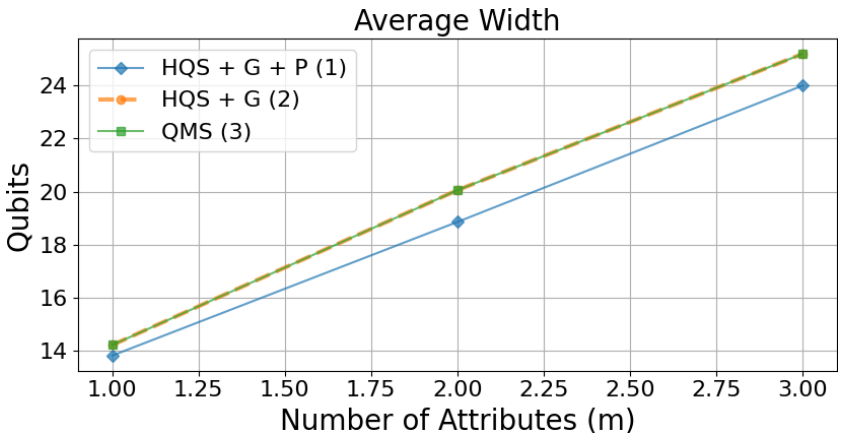


Figure 4.6: Average Width as m increases

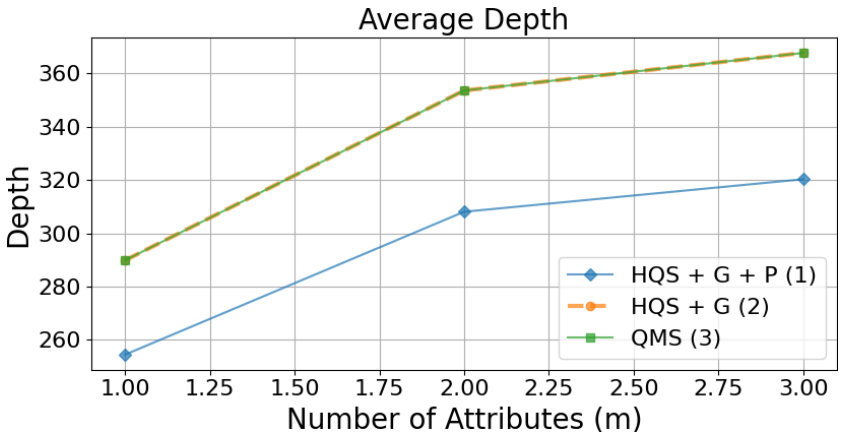


Figure 4.7: Average Depth as m increases

As m attributes scales, our proposed method (1) once again shows reduced width and depth compared to methods (2) and (3), highlighting the cost efficiency of implementing pruning with the quantum search algorithm.

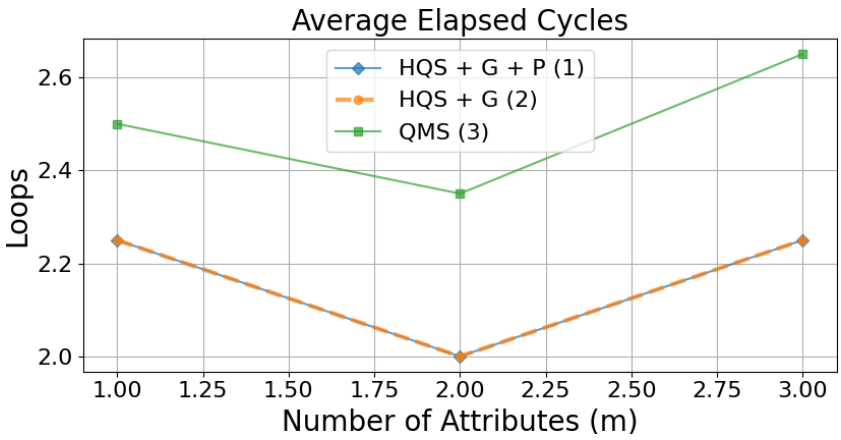


Figure 4.8: Average Elapsed Cycles as m increases

While method (3) still loses in elapsed cycles, methods (1) and (2) show similar results here. Once again, this may be due to the relatively limited sample size we were able to test with. Testing for more instances and bigger problems size may provide clearer discrepancy between methods.

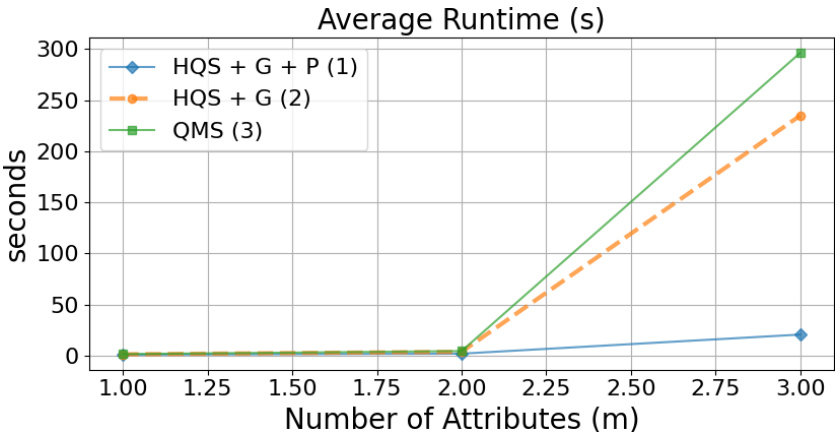


Figure 4.9: Average Runtime as m increases

Concerning the simulation runtime of our second set of experiments, we see a similar trend as our first set of experiments. Method (1) shows highest efficiency with the slowest growth in runtime, followed by (2), and finally (3). Methods (2) and (3) both exhibit a sharp increase in runtime at $m = 3$, with (3) demonstrating the highest jump and (2) following close behind. In contrast, our proposed method of (1) shows a significantly smoother and gradual increase in runtime, maintaining lower values across all tested instances. Although we could only test for m upwards of (3) due to limited resources, the trend suggests that our proposed method has higher potential scalability for problems of larger size.

Chapter 5

Conclusion

We proposed a hybrid quantum-classical algorithm and oracle circuit design for the MDKP. This problem, characterized by its strongly NP-hard complexity, poses significant challenges in finding exact solutions efficiently as problem size increases. To find exact solutions in the MDKP more efficiently, we use a hybrid quantum search with classical pre-processing techniques.

Our method demonstrated superior efficiency in terms of depth, width, elapsed cycles, and runtime when comparing to algorithm variants that use pre-existing methods. Furthermore, our method's reduction in growth of elapsed cycles and runtime as the size of the problem increases indicates its potential for scalability. Although instances with simultaneous growth in both n items and m attributes were not tested, our results strongly suggest that our method would exhibit even greater efficiency and cost benefits in such scenarios.

Current approaches to solving the MDKP rely heavily on classical algorithms which use heuristics to find approximate solutions, as finding exact solution face huge limitations in scalability and efficiency. While these heuristic methods provide practical solutions within reasonable time frames and cost, they lack the ability to guarantee the best solution. Although the difference between approximate and exact solutions may be small, this limitation can have significant implications in scenarios where 1-2% precision is crucial. Our proposed method leverages the benefits of classical heuristics while refining the solutions with quantum methods. This hybrid approach not only provides solutions to the MDKP but also offers a framework for solving similar optimization problems.

However, there is room for further optimization. Future work may involve circuit optimization technique such as making a more efficient integer comparator for our current use case. Additionally, exploring the algorithm's performance with problems of larger size and scenarios with simultaneous growth in n and m would provide greater insight into the scalability and performance of our method.

References

[1] Akçay, Y., Li, H., and Xu, S. H. Greedy algorithm for the general multidimensional knapsack problem. *Annals of Operations Research 150* (2007), 17–29.

[2] Awasthi, A., Bär, F., Doetsch, J., Ehm, H., Erdmann, M., Hess, M., Klepsch, J., Limacher, P. A., Luckow, A., Niedermeier, C., et al. Quantum computing techniques for multi-knapsack problems. In *Science and Information Conference*, Springer (2023), 264–284.

[3] Boyer, M., Brassard, G., Høyer, P., and Tapp, A. Tight bounds on quantum searching. *Fortschritte der Physik 46*, 4–5 (June 1998), 493–505.

[4] Chu, P. C., and Beasley, J. E. A genetic algorithm for the multidimensional knapsack problem. *Journal of heuristics 4* (1998), 63–86.

[5] Cuccaro, S. A., Draper, T. G., Kutin, S. A., and Moulton, D. P. A new quantum ripple-carry addition circuit, 2004.

[6] Devitt, S. J., Munro, W. J., and Nemoto, K. Quantum error correction for beginners. *Reports on Progress in Physics 76*, 7 (June 2013), 076001.

[7] Draper, T. Addition on a quantum computer.

[8] Durr, C., and Hoyer, P. A quantum algorithm for finding the minimum, 1999.

[9] Furusawa, A., Sørensen, J. L., Braunstein, S. L., Fuchs, C. A., Kimble, H. J., and Polzik, E. S. Unconditional quantum teleportation. *Science 282*, 5389 (Oct. 1998), 706–709.

[10] Gisin, N., Ribordy, G., Tittel, W., and Zbinden, H. Quantum cryptography. *Rev. Mod. Phys. 74* (Mar 2002), 145–195.

[11] Grover, L. K. A fast quantum mechanical algorithm for database search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (1996), 212–219.

[12] Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C. J., Lishman, J., Gacon, J., Martiel, S., Nation, P. D., Bishop, L. S., Cross, A. W., Johnson, B. R., and Gambetta, J. M. Quantum computing with Qiskit, 2024.

[13] Kellerer, H., Pferschy, U., and Pisinger, D. *Knapsack Problems*. Springer, 2004.

[14] Kellerer, H., Pferschy, U., Pisinger, D., Kellerer, H., Pferschy, U., and Pisinger, D. *Multidimensional knapsack problems*. Springer, 2004.

[15] Kragh, H. *Paul Dirac and The Principles of Quantum Mechanics*. Max-Planck-Gesellschaft zur Förderung der Wissenschaften, 2013.

[16] Lai, X., Hao, J., Yue, D., and Gao, H. A diversification-based quantum particle swarm optimization algorithm for the multidimensional knapsack problem. In *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, IEEE (2018), 315–319.

[17] Mathews, G. B. On the Partition of Numbers. *Proceedings of the London Mathematical Society s1-28*, 1 (11 1896), 486–490.

[18] Meter, R. V., Munro, W. J., Nemoto, K., and Itoh, K. M. Arithmetic on a distributed-memory quantum multicomputer. *ACM Journal on Emerging Technologies in Computing Systems 3*, 4 (Jan. 2008), 1–23.

[19] Shor, P. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), 124–134.

[20] Shor, P. Introduction to quantum algorithms. *AMS Proceedings of Symposium in Applied Mathematics 58* (05 2000).

[21] Van Meter, R., and Itoh, K. M. Fast quantum modular exponentiation. *Physical Review A 71*, 5 (May 2005).

[22] Wootters, W. K., and Zurek, W. H. A single quantum cannot be cloned. *Nature 299*, 5886 (Oct. 1982), 802–803.