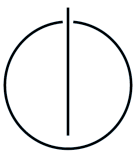
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Information Systems

# Iterative Quantum Optimization Algorithms for the Knapsack Problem

Nico Stabla



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

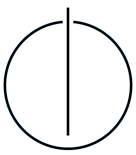
TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Information Systems

# Iterative Quantum Optimization Algorithms for the Knapsack Problem

# Iterative Quantenoptimierungsalgorithmen für das Knapsack Problem

| | |
|---|---|
| Author: | Nico Stabla |
| Examiner: | Prof. Dr. Christian Mendl |
| Supervisor: | Jernej Rudi Finžgar, M.Sc. |
| Submission Date: | 10.12.2024 |



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 10.12.2024 Nico Stabla

# Abstract

We propose and implement a family of iterative quantum optimization algorithms tailored to the Knapsack Problem (KP), a combinatorial optimization problem with applications in logistics, supply chain management, finance, and resource allocation. Efficiently solving KP is critical for maximizing resource utilization and dynamically adapting to constraints, especially in scenarios where classical methods face challenges in scalability and real-time responsiveness.

Our approach leverages the Quantum Approximate Optimization Algorithm (QAOA) to extract correlations from quantum states, providing probabilistic insights into the likelihood of items contributing to the optimal solution. These correlations inform problem-specific update steps in an iterative process to simplify the problem.

We benchmark our algorithms against classical heuristics, such as the Greedy algorithm, across general and special scenarios using quantum simulation. General cases utilize 12 to 24 qubits and are created randomly within that boundary. Special cases utilize 12 qubits and focus on challenging configurations in that the Greedy algorithm fails to find the optimal solution, such as misleading value-to-weight ratios. Our results demonstrate that the solution quality improves with increasing circuit depth (up to 3) in many cases and showcase the competitiveness of our algorithms with classical heuristics, particularly in complex scenarios. While all of our algorithms are able to improve their solution quality for higher depths $p$, one of our algorithms fails to improve upon solution quality for higher depths in general cases.

This work positions iterative quantum optimization algorithms as a framework for solving the KP, with the promise of further scalability and practicality as hardware advances.

# Abbreviations

**BnB** Branch-and-Bound

**DP** Dynamic Programming

**KP** Knapsack Problem

**QAOA** Quantum Approximate Optimization Algorithm

**QIRO** Quantum Informed Recursive Optimization

**QUBO** Quadratic Unconstrained Binary Optimization

**RQAOA** Recursive Quantum Approximate Optimization Algorithm

# Contents

# 1 Introduction

In this introduction chapter, we will present the importance of quantum optimization and the Knapsack Problem (KP). We also explain related or similar approaches to ours in the related work section.

## 1.1 Motivation

Many combinatorial optimization problems are known to be NP-Hard, meaning their solution times increase exponentially with problem size [1]. This computational bottleneck makes it challenging to solve large instances within practical timeframes. In fields like logistics, finance, telecommunications, and healthcare, these optimization problems arise frequently and require near-instantaneous solutions to adapt to changing conditions.

For example, consider a parcel delivery company in Munich. The company assigns one hundred thousand parcels to a hundred drivers each day, but if a driver has an accident mid-shift, the company must quickly reallocate parcels among the remaining drivers to maintain punctual deliveries. In such scenarios, time constraints add complexity to the optimization problem, requiring both speed and flexibility. Whereas this example requires many optimization algorithms, KP plays a significant role in optimally packing and distributing the parcels among the drivers that spontaneously drive to the location of the accident.

Traditional computing methods, while effective for smaller instances, struggle with large, real-time optimization tasks. Methods such as Dynamic Programming (DP) [2] or Branch-and-Bound (BnB) [3] often cannot provide timely solutions for very large-scale problems, resulting in reduced operational flexibility. Quantum computers might be able to solve certain large optimization problems faster to optimality than traditional methods. In this work, we try to understand, if iterative quantum algorithms are able to solve the KP to optimality.

This thesis explores iterative quantum algorithms for solving the KP, building on advancements in quantum optimization algorithms. Specifically, it extends previous approaches such as the QAOA [4] and Recursive Quantum Approximate Optimization Algorithm (RQAOA) [5]. The iterative algorithms proposed by Finžgar *et al.* [6] and

Brady *et al.* [7], which have shown promise in tackling other NP-hard problems (e.g., the Maximum Independent Set problem), serve as a foundation for this work.

## 1.2 Related Work

Quantum computing has shown potential in addressing complex combinatorial optimization problems such as the KP. Existing approaches can be divided into these categories: quantum annealing, extensions of QAOA, quantum-enhanced classical techniques, and quantum tree generator.

### Quantum Annealing for the Knapsack Problem

Quantum annealing, implemented on hardware such as D-Wave quantum computers, has been investigated as a potential solution for NP-hard problems like the KP. Bożejko *et al.* [8] demonstrated the feasibility of solving the binary KP using quantum annealing, leveraging a Branch-and-Bound solver to optimize solutions. Their work highlights the ability of quantum annealers to solve Quadratic Unconstrained Binary Optimization (QUBO) formulations.

Unlike Bożejko *et al.*, our work relies on iteratively solving and simplifying the QUBO problem. Besides, we limit our work to simulated quantum devices on classical hardware.

### Quantum Approximate Optimization Algorithm Extensions

Extensions to the standard QAOA have been proposed to improve its performance on constrained optimization problems like the KP. Van Dam *et al.* [9] introduced xQAOA, which integrates preconfigured initial states generated through classical heuristics, such as greedy algorithms, and employs custom mixer Hamiltonians.

Similarly, Awasthi *et al.* [10] explored warm-started QAOA for the multi-KP, leveraging classical preprocessing to initialize the quantum circuit.

Our work does not warm-start QAOA, but iteratively applies QAOA to solve the KP. However, warm-starting QAOA marks a future work of our algorithms.

### Iterative Quantum Optimization Algorithms

Iterative quantum optimization approaches, such as Quantum Informed Recursive Optimization (QIRO) and MinQ, have been developed by Finžgar *et al.* [6] and Brady *et al.* [7], respectively. These algorithms iteratively refine solutions by leveraging quantum correlations from low-energy states to simplify the problem space progressively. The

quantum correlations are calculated using QAOA. While originally designed for problems like Maximum Independent Set, they lay a strong foundation for iterative methods in quantum optimization.

Because these methods are not specifically tailored to the KP, their iterative framework serves as a foundation to this work.

**Quantum Tree Generator for the Knapsack Problem**

Wilkening *et al.* [11] introduced a novel quantum tree generator that constructs all feasible solutions for the binary KP in quantum superposition. This technique offers exponential memory savings.

The primary connection to our work lies in addressing the KP.

# 2 Background

In the following chapter, we explain concepts that are required to understand this work. We start with defining QUBO problems [12] that we need to solve the KP using quantum technologies. Then, we define the KP and reformulate it so that we can solve it using quantum devices. Besides, we introduce important classical solvers to the KP and summarize the computational complexity of classical optimal solvers. Lastly, we introduce quantum algorithms that are relevant to our work. The foundation lays QAOA [4] and iterative quantum approaches such as QIRO [6] and MinQ [7].

## 2.1 Quadratic Unconstrained Binary Optimization Problem

Quadratic Unconstrained Binary Optimization (QUBO) is an NP-hard optimization problem that represents a wide variety of combinatorial optimization tasks [12]. The objective of a QUBO problem is to find the binary vector $x \in \{0,1\}^N$ that minimizes a quadratic polynomial over binary variables. It can be expressed as:

$$f_Q(x) = x^T Q x = \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{ij} x_i x_j$$

where $x_i \in \{0,1\}$ are binary decision variables, and $Q$ is an $n \times n$ matrix that encodes the quadratic relationships between the variables [12]. The diagonal terms in the matrix represent one-point correlations, while the off-diagonal terms in the matrix represent two-point correlations. The goal is to determine the values of the binary variables $x_i$ that minimize this quadratic expression.

#### Relation to the Ising Model

The Ising model, originally formulated in statistical physics to describe the behavior of spins in a magnetic system, is closely related to QUBO. The key difference lies in the domain of the variables: QUBO uses binary variables $x_i \in \{0,1\}$ , while the Ising model uses spin variables $s_i \in \{-1,+1\}$ [13]. Mapping binary variables to Ising variables, we can represent a QUBO problem as a Ising model. Quantum annealers are able to find the ground state of Ising models [14]. The ground state is the lowest energy state

of a quantum system and represents the binary vector $x \in \{0,1\}^N$ that minimizes a quadratic polynomial over binary variables. In short: the ground state represents the optimal solution of a QUBO problem.

## 2.2 Knapsack Problem

The Knapsack Problem (KP) is one of the most extensively studied and significant NP-Hard optimization problems, with numerous real-world applications such as cryptography [15]. It is recognized as the simplest form of a maximization problem [16]. In this section, the problem will be formally defined and translated into a Hamiltonian that represents the QUBO problem in Ising form.

### 2.2.1 Definition

Given $N$ items labeled by $\alpha$, each with an integer weight $w_\alpha$ and an integer value $c_\alpha$, the objective is to select a subset of these items such that the total weight does not exceed a specified maximum capacity $W$, and the total value is maximized. We introduce a binary variable $x_\alpha$, where $x_\alpha = 1$ if item $\alpha$ is included in the knapsack and $x_\alpha = 0$ otherwise. Each item can be included at most once. [16]

The mathematical formulation of the KP is presented below:

$$\begin{aligned} \max. \quad & \sum_{\alpha=1}^{N} c_\alpha x_\alpha \\ \text{subject to} \quad & \sum_{\alpha=1}^{N} w_\alpha x_\alpha \leq W \\ & x_\alpha \in \{0,1\} \quad \forall \alpha = 1, \ldots, N. \end{aligned}$$

### 2.2.2 Translation into a Quadratic Unconstrained Binary Optimization Problem

As mentioned in 2.1, a QUBO problem can be formulated as an Ising model. Furthermore, quantum annealers are able to find the ground state of Ising models, which means that it can find its minimal state. Thus, a quantum annealer can find the optimal solution of a QUBO problem. Because we want to find the optimal solution of the KP using quantum devices, we formulate the KP into a QUBO problem with Ising variables. From Lucas *et al.* [13], we know that the KP can be translated into a QUBO problem.

For the translation of a KP to a QUBO problem, we introduce Ising variables, $y_n = \pm 1$ indicating if the total weight of the knapsack is exactly $n$ ($1 \leq n \leq W$). Furthermore, we perform the mapping $z_\alpha = 2x_\alpha - 1$ to Ising variables $z_\alpha = \pm 1$. The Hamiltonian $H = H_A + H_B$ represents the QUBO problem in Ising form. $H_A$ ensures constraint enforcement so that the weight of the items in the knapsack match the claimed weight and that the weight can only take on one value [13].

Based on Lucas *et al.* [13], we get:

$$H_A = A\left(1 - \sum_{n=1}^{W} y_n\right)^2 + A\left(\sum_{n=1}^{W} n y_n - \sum_{\alpha=1}^{N} w_\alpha z_\alpha\right)^2. \tag{2.1}$$

The second part $H_B$ of the Hamiltonian ensures objective maximization so that the total value of the knapsack is maximized [13]. Formally:

$$H_B = -B \sum_{\alpha=1}^{N} c_\alpha z_\alpha. \tag{2.2}$$

Parameters $A$ and $B$ can be fine-tuned for each problem to increase solution quality. $0 < B \max(c_\alpha) < A$ applies [13].

**Translation into a Matrix**

We want to represent the Hamiltonian $H = H_A + H_B$ as a matrix that represents the energy of the system in order to find the low-energy state.

In the following, we translate all $z_\alpha$, $w_\alpha$ and $y_n$ into corresponding vectors, and we introduce the vector $\vec{a}$ to ensure dimensionality.

For $H_A$, we start with Equation 2.1:

$$\begin{aligned}
H_A &= A\left(1 - \sum_{n=1}^{W} y_n\right)^2 + A\left(\sum_{n=1}^{W} n y_n - \sum_{\alpha=1}^{N} w_\alpha z_\alpha\right)^2 \\
&= A\left(1 - \sum_{n=1}^{W} a_n y_n\right)^2 + A\left(\sum_{n=1}^{W} n y_n - \sum_{\alpha=1}^{N} w_\alpha z_\alpha\right)^2 \\
&= A(1 - \vec{a}^T\vec{y})^2 + A(\vec{n}^T\vec{y} - \vec{w}^T\vec{z})^2 \\
&= A(1 - 2\vec{a}^T\vec{y} + \vec{y}^T(\vec{a}\vec{a}^T)\vec{y}) + A(\vec{y}^T(\vec{n}\vec{n}^T)\vec{y} - 2\vec{y}^T(\vec{n}\vec{w}^T)\vec{z} + \vec{z}^T(\vec{w}\vec{w}^T)\vec{z})
\end{aligned}$$

$$\text{with } \vec{a} = (1, ..., 1)^T, \vec{n} = (1, ..., W)^T.$$

For $H_B$, we start with Equation 2.2:

$$\begin{aligned} H_B &= -B \sum_{\alpha=1}^{N} c_\alpha z_\alpha \\ &= -B(\vec{c}^T \vec{z}). \end{aligned}$$

## 2.3 Classical Solvers for the Knapsack Problem

In the following, classical approaches to solve the KP are introduced that we later use to compare our algorithms with. First, we introduce the Greedy and Advanced Greedy algorithm. Then, we introduce DP and BnB. There also exist many more classical solvers that we will not investigate in this work. In the end of this section, we summarize the computational complexity of optimal solvers.

### 2.3.1 Greedy Algorithm

The Greedy algorithm addresses the KP by prioritizing items based on their value-to-weight ratios [17].

The procedure involves the following steps:

1. **Calculation of value-to-weight ratio**: For each item $\alpha$, compute the value-to-weight ratio given by $\frac{c_\alpha}{w_\alpha}$.
2. **Sorting of items**: Sort the items in descending order according to their calculated value-to-weight ratios.
3. **Selection of items**: Sequentially add items to the knapsack, starting with those having the highest value-to-weight ratios. Continue this process until the total weight of the selected items reaches or approaches the maximum weight constraint $W$.

By systematically selecting items that provide the greatest value per unit of weight, the Greedy algorithm efficiently constructs a solution with a worst-case runtime of $O(n \log n)$. However, it does not guarantee to find the optimal solution.

### 2.3.2 Advanced Greedy Algorithm

The Advanced Greedy algorithm was introduced by Glover [18] and extends the conventional Greedy approach by integrating the following key steps:

1. **Initialization**:
   - Set the remaining capacity of the knapsack RHS to the maximum allowable weight $W$.
   - Initialize an empty solution list, to store the indices of selected items.
2. **Sorting and labeling**:
   - Sort the items in ascending order based on their weights. This facilitates the identification of items that occupy the least space, potentially allowing for the inclusion of more items within the weight constraint.
   - Label the items with $j$ starting with the first item of the list.
3. **Capacity-based allocation**:
   - For each item, calculate the maximum number of times it can fit into the remaining capacity RHS, denoted as $n_j = \left\lfloor \frac{\text{RHS}}{w_j} \right\rfloor$.
   - Determine $n_0$, the largest index such that the cumulative weight of the smallest items does not exceed RHS.
4. **Priority metric calculation**:
   - Compute a priority metric $P_{Kj}$ for each item, defined as:

     $$P_{Kj} = c_j \times \min(n_j, n_0)$$

     where $c_j$ is the value of item $j$. This metric balances the item's value with its potential contribution based on the remaining capacity and the distribution of weights.
5. **Selection of items**:
   - Sort the items in descending order based on their calculated $P_{Kj}$ values.
   - Iteratively select items from the sorted list, adding an item to the solution if its weight does not exceed the remaining capacity RHS. After selection, update RHS, total value, and total weight.

The Advanced Greedy algorithm does not guarantee to find the optimal solution.

### 2.3.3 Dynamic Programming

Dynamic Programming (DP) is a robust algorithmic paradigm for solving complex optimization problems [2]. Its key principle is solving each sub-problem once and

storing the result. For DP to work, a problem must have a solution that can be built from optimal solutions of its sub-problems, a property known as optimal substructure. Due to the optimal substructure, DP guarantees to find the optimal solution. The KP fulfills this requirement [2].

**Algorithmic Approach**

Based on Kellerer *et al.* [2], the DP approach for the KP involves the following steps:

1. **Problem decomposition**: Divide the original KP into smaller sub-problems. Each sub-problem represents the decision of whether to include an item $\alpha$ in the knapsack given a remaining weight capacity $W$.
2. **Recursive relation**: Establish a recursive relationship that relates the solution of a sub-problem to the solutions of its smaller sub-problems. The recursive relation can be defined as:

$$\text{Knap}(i, w) = \max \begin{cases} c_i + \text{Knap}(i-1, w-w_i), & \text{if } w_i \leq w \\ \text{Knap}(i-1, w), & \text{otherwise} \end{cases}$$

   where $\text{Knap}(i, w)$ represents the maximum value achievable with the first $i$ items and a weight capacity $w$, $c_i$ is the value of item $i$, and $w_i$ is its weight.
3. **Memorization**: Store the solutions of the sub-problems in a table.
4. **Iterative computation**: Fill the table iteratively based on the recursive relation.
5. **Solution construction**: Reconstruct the optimal solution by tracing back through the table to determine which items were included in the knapsack.

### 2.3.4 Branch-and-Bound

Branch-and-Bound (BnB) solves problems based on an intelligent complete enumeration of the solution with the guarantee that the parts not considered for optimality are no subsets of the optimal solution [3]:

- **Branching**: The solution set is divided into smaller subsets, repeatedly, until each subset contains a single feasible solution. Through considering all possible solutions, the best global solution is picked and thus BnB guarantees to find the optimal solution.
- **Bounding**: Upper and lower bounds are calculated for each subset of the solution space. A lower bound may be the best solution found so far, and an upper bound exceeds the problem constraints.

In this work, we will not further investigate Branch-and-Bound because we focus on small instances that DP can solve to optimality fast.

### 2.3.5 Computational Complexity of Classical Solvers

For the KP no polynomial time algorithm is known for computing its optimal solution and there is evidence that none exists [16]. However, the most efficient classical solvers for the KP have a pseudo-polynomial runtime complexity [19]. This means that the running time is bounded by problem size and one or several other input variables. The DP for the KP is bounded by the size of the item set and the knapsack capacity in worst-case scenarios, in short: $O(NW)$ [2]. Also, we want to note that BnB has a non-polynomial worst-case runtime of $O(2^N)$ meaning that the runtime grows exponentially. Please note that approximation algorithms exist that are polynomial but do not guarantee to find the optimal solution [20].

## 2.4 Quantum Optimization Algorithms

In this section, we introduce important quantum optimization algorithms that serve as a foundation to this work. We start with the QAOA [4], then we describe Iterative Quantum Algorithms. QAOA serves as a foundation to Iterative Quantum Algorithms by calculating necessary quantum correlations between qubits.

### 2.4.1 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical algorithm designed to solve combinatorial optimization problems by approximately finding the lowest energy state of a quantum system. As described by Farhi *et al.* [4], the algorithm combines quantum evolution and classical optimization to find an optimal solution by alternating between quantum and classical computation: the quantum computer evaluates the energy for a given set of parameters $(\beta, \gamma)$, and the classical optimizer adjusts the parameters to minimize the energy.

The QAOA is a local algorithm, meaning that qubits can only interact if they are connected by less than a certain distance in the interaction graph that represents the optimization problem.

In this context, the problem is formulated as finding the ground state (the lowest energy state) of a cost Hamiltonian $H_c$. For example, when the problem is expressed in a QUBO form, the cost Hamiltonian is given as:

$$H_c = \sum_{i,j} J_{ij} \hat{Z}_i \hat{Z}_j + \sum_i h_i \hat{Z}_i$$

where:

- $\hat{Z}_i$ are the Pauli-Z operators acting on qubit $i$,
- $J_{ij}$ represents the interaction between qubits $i$ and $j$,
- $h_i$ represents the linear bias on qubit $i$.

The mixer Hamiltonian $H_{mix}$ is introduced to enable exploration of the solution space by allowing transitions between different quantum states. It is defined as:

$$H_{mix} = -\sum_{i=1}^{n} \hat{X}_i$$

where $\hat{X}_i$ are the Pauli-X operators, which flip the state of qubit $i$. The mixer Hamiltonian helps to explore different configurations and avoid getting stuck in local minima by flipping the qubits, creating a mixing effect.

The intuition of the QAOA algorithm is that it alternates between applying the cost Hamiltonian $H_c$, which drives the system toward lower energy solutions, and the mixer Hamiltonian $H_{mix}$, which promotes exploration of the solution space.

**Quantum Approximate Optimization Algorithm Circuit and Quantum State Preparation**

The initial state of the system, denoted as $\psi_0$, is a uniform superposition over all possible bit strings:

$$|\psi_0\rangle = |+\rangle^{\otimes n}$$

where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and $n$ is the number of qubits. This state is easy to prepare and gives equal probability to all potential solutions.

For a given depth $p$, the quantum state produced by QAOA is given by:

$$|\psi(\beta, \gamma)\rangle = e^{-i\beta_p H_{mix}} e^{-i\gamma_p H_c} \dots e^{-i\beta_1 H_{mix}} e^{-i\gamma_1 H_c} |\psi_0\rangle .$$

This alternating sequence of applying the mixer Hamiltonian and the cost Hamiltonian creates a quantum circuit of depth $p$, where the parameters $\gamma_1, \dots, \gamma_p$ determine how long the cost Hamiltonian $H_c$ is applied during each layer, while $\beta_1, \dots, \beta_p$ determine how long the mixer Hamiltonian $H_{mix}$ is applied during each layer.

#### Classical Optimization of Parameters

The parameters $(\beta, \gamma)$ are optimized using a classical routine. The goal of the classical optimizer is to minimize the expectation value of the cost Hamiltonian in the quantum state $|\psi(\beta, \gamma)\rangle$, which corresponds to finding the best approximation of the ground state. This optimization is formalized as:

$$\min_{\beta,\gamma} \langle \psi(\beta, \gamma)| H_c |\psi(\beta, \gamma)\rangle .$$

This expectation value represents the average energy of the quantum state, and the classical optimizer searches for the parameter set $(\beta, \gamma)$ that minimizes this energy.

The algorithm alternates between quantum and classical computation: the quantum computer evaluates the energy for a given set of parameters $(\beta, \gamma)$, and the classical optimizer adjusts the parameters to minimize the energy.

#### Recursive Quantum Approximate Optimization Algorithm

There also exists a recursive approach to QAOA proposed by Bravyi *et al.* [5], known as RQAOA, which simplifies the problem at each recursion step by solving a smaller, reduced version of the original problem. At each step, RQAOA prepares the QAOA state for that iteration and minimizes the cost function. The RQAOA is a non-local algorithm, meaning that the distances between nodes decrease with each iteration as other nodes are removed. Thus, it allows communications between more qubits in the interaction graph that represents the optimization problem than QAOA.

### 2.4.2 Iterative Quantum Algorithms

Finžgar *et al.* [6] and Brady *et al.* [7] proposed iterative variations of the QAOA that incorporate quantum information from low-energy quantum states to guide the classical optimization process in a problem-specific manner. Unlike the RQAOA approach, these methods make problem-specific updates. The iterative variations are motivated by Greedy algorithms and aim to perform similar for low depth $p = 1$ but are expected to return higher solution qualities for higher depths $p > 1$. There are four different algorithms that differ in problem simplification: QIRO, MinQ, MaxQ and MMQ. We will start with the general procedure of those algorithms, which is the same. Then, we introduce the differences in algorithm specific simplification strategies.

#### General Procedure

Repeat the following as long as the problem size is larger than 0:

1. Create a problem specific quantum Hamiltonian $H$.
2. Prepare a low-energy quantum state $|\psi\rangle$ using QAOA.
3. Store the correlations of $|\psi\rangle$ in a Matrix $M$.
4. Perform problem and algorithm specific simplification based on $M$.

**Meaning of One-Point and Two-Point Correlations**

The quantum state $|\psi\rangle$ is used to extract information. This information comes in the form of:

- **One-point correlations** $\langle\psi|\hat{Z}_i|\psi\rangle$, which measure the expected value of the Pauli-$Z$ operator for qubit $i$, indicating whether the qubit tends to be in the $|0\rangle$ or $|1\rangle$ state. One-point correlations are stored diagonally in matrix $M$.
- **Two-point correlations** $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle$, which capture the correlations between pairs of qubits $i$ and $j$. These indicate whether two qubits are likely to be in the same or opposite states. Two-point correlations are stored off-diagonally in matrix $M$.

**Algorithm Specific Simplification**

- **QIRO**: Sort the correlations in $M$ in ascending order and consider one- and two-point correlations. Pick the highest correlation entry. If it is a one-point correlation, and it is positive, add the corresponding entry to the solution. If it is a two-point correlation, perform simplification based on both entries, e.g., merge or delete entries. In every case, delete it from the problem.
- **MinQ**: Sort the correlations in $M$ in ascending order and only consider one-point correlations. First, consider positive correlations (intuition: first take a look at items that positively contribute to the optimal solution). Pick the highest correlation entry and add it to the solution if it is positive. Delete it from the problem.
- **MaxQ**: Sort the correlations in $M$ in descending order and only consider one-point correlations. First, consider negative correlations (intuition: first take a look at items that negatively contribute to the optimal solution). Pick the highest correlation entry and add it to the solution if it is positive. Delete it from the problem.

- **MMQ**: Sort the correlations in $M$ in ascending order and only consider one-point correlations. Pick the highest correlation entry and add it to the solution if it is positive. Delete it from the problem.

# 3 Quantum Enhanced Algorithms

In this chapter, we present four quantum-enhanced algorithms specifically designed for optimizing the KP. Our algorithms employ iterative update rules, which progressively simplify the KP by leveraging quantum correlations to guide the selection of items.

This chapter first outlines the general steps underlying these algorithms, followed by a detailed procedural explanation. Last, we will introduce other approaches that we tested during our implementation phase.

## 3.1 General Procedure

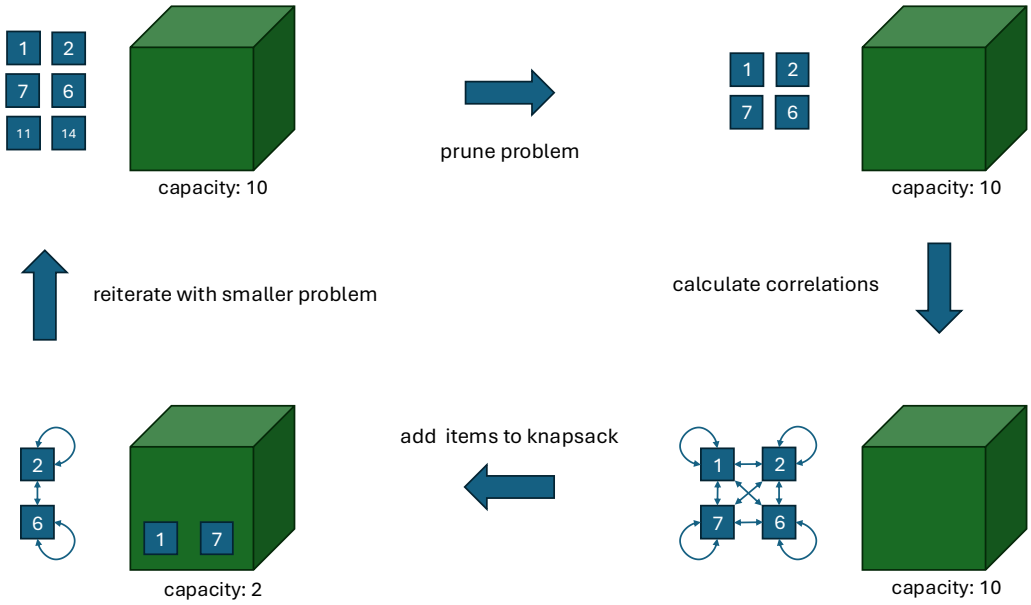



Figure 3.1: Illustration of the item selection procedure for the KP using our algorithms. The blue rectangles represent items, with their weight and value. For simplicity, value and weight are equal. The green box represents the knapsack subject to capacity constraints.

In Figure 3.1, we provide an overview of the typical workflow. Initially, items exceeding the weight constraint are excluded through pruning. Next, we compute one- and two-point correlations using the QAOA. These correlations inform our update rules, determining item inclusion in or exclusion from the knapsack. In that way, we decrease the problem size with each iteration. We repeat the previously mentioned steps as long as there still are items in the problem left that fit into the knapsack.

## 3.2 Detailed Procedure

Our algorithms solve the KP by iteratively pruning infeasible items, formulating the problem as a quantum Hamiltonian, preparing quantum states, and applying update rules based on computed quantum correlations. We will introduce necessary steps here and put them together in the end in Algorithm 1.

### 3.2.1 Problem Creation

The KP is created with a weight constraint $W$, and a set of items $\alpha$, each with weight $w_\alpha$ and value $c_\alpha$. Items exceeding the weight constraint $W$ are pruned. Specifically, for each item $\alpha$, if $w_\alpha > W$, it is excluded from consideration. Fine-tuning parameters $A$ and $B$ are set according to Equations 2.1 and 2.2, and the problem is reformulated into a quantum Hamiltonian $H$ (based on 2.2.2).

### 3.2.2 Quantum State Preparation

Using the pruned Hamiltonian $H$, an initial quantum state $|\psi_0\rangle = |+\rangle^{\otimes n}$ is prepared. Then, the low-energy state $|\psi\rangle$ is calculated using QAOA (as explained in 2.4.1). This process computes one-point correlations $\langle\psi|\hat{Z}_i|\psi\rangle$ and two-point correlations $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle$.

### 3.2.3 Update Rules

The quantum correlations inform update rules for including or excluding items. The rules vary based on one- or two-point correlations. Please note that other rules are also possible and might achieve a higher solution quality for some instances.

#### One-Point Correlations

One-point correlations $\langle\psi|\hat{Z}_i|\psi\rangle$ can be interpreted as follows:

- $\langle\psi|\hat{Z}_i|\psi\rangle > 0$: Item $i$ positively contributes to optimization.

- $\langle\psi|\hat{Z}_i|\psi\rangle < 0$: Item $i$ negatively contributes to optimization.
- $\langle\psi|\hat{Z}_i|\psi\rangle = 0$: Item $i$ has a neutral effect.

The corresponding update rules are:

- Include item $i$ in $S$ if $\langle\psi|\hat{Z}_i|\psi\rangle > 0$ and $w_\alpha \leq W$.
- Exclude item $i$ if $\langle\psi|\hat{Z}_i|\psi\rangle < 0$.
- Delete item $i$ if $\langle\psi|\hat{Z}_i|\psi\rangle = 0$.

**Two-Point Correlations**

Two-point correlations $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle$ describe relationships between items:

- $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle > 0$: Including item $i$ (or $j$) increases the likelihood of item $j$ (or $i$) positively contributing to the optimal solution.
- $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle < 0$: Negative relationship; including one decreases the likelihood of the other contributing to the optimal solution.
- $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle = 0$: Neutral relationship between $i$ and $j$.

The update rules are:

- For $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle > 0$: Include the item with the higher one-point correlation if it fits in the knapsack.
- For $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle < 0$: Delete the item with the lower one-point correlation.
- For $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle = 0$: Prioritize the item with the higher one-point correlation and delete the lower one-point correlation item.

### 3.2.4 Algorithm-Specific Procedures

Each algorithm prioritizes items based on specific rules:

- **QIRO**: Pick the highest two- or one-point correlation entry based on QAOA. Perform the update rules on the picked correlation item.
- **MinQ**: First consider positive correlations. Pick the highest one-point correlation entry based on QAOA and perform update rules on it.
- **MaxQ**: First consider negative correlations. Pick the lowest one-point correlation entry based on QAOA and perform update rules.
- **MMQ**: Pick the highest one-point correlation entry based on QAOA and perform update rules on it.

### 3.2.5 General Algorithm Workflow

The overall workflow is shown in Algorithm 1.

**Algorithm 1** Iterative Quantum Optimization Algorithm for the KP

**Require:** A set of items $\alpha$ with weights $w_\alpha$ and values $v_\alpha$, and a maximum weight constraint $W$

**Ensure:** Solution set $S$ containing selected items

1: Initialize the solution set $S = \emptyset$
2: **while** $\exists\, \alpha$ with $w_\alpha \leq W$ **do**
3: Prune items: exclude any item $\alpha$ for which $w_\alpha > W$
4: Formulate the pruned problem as a quantum Hamiltonian $H$
5: Prepare initial quantum state $|\psi_0\rangle = |+\rangle^{\otimes n}$
6: Apply QAOA to obtain a low-energy quantum state $|\psi(\beta, \gamma)\rangle$
7: Store the one- and two-point correlations $\langle\psi|\hat{Z}_i|\psi\rangle$ and $\langle\psi|\hat{Z}_i\hat{Z}_j|\psi\rangle$ in matrix $M$
8: Apply algorithm specific simplifications based on the correlation matrix $M$
9: Update $W$ based on items added to $S$ in this iteration
10: **end while**
11: **return** Solution set $S$ containing the final selection of items

## 3.3 Other Tested Approaches

In this section, we introduce approaches that we tried out during the implementation phase of this work. We investigated all of those approaches and decided not to follow them more in detail. The approaches include: ignoring slack variables to lower the required number of qubits, a very greedy quantum algorithm that only requires one iteration and, different update rules.

### 3.3.1 Ignoring Slack Variables

We found that slack variables $y_1, \ldots, y_W$ can be omitted by directly (classically) encoding the weight constraint. This was achieved by validating whether an item's weight $w_\alpha$ satisfies the remaining capacity condition, $w_\alpha \leq W$. For consistency, we included slack variables to our implementation.

### 3.3.2 Very Greedy Quantum Algorithm

In our efforts to accelerate the solution of the KP using quantum technology, we proposed an approach that computes the quantum correlations once, sequentially adding items with the highest correlation values to the solution until the knapsack capacity is reached. However, Fischer *et al.* [21] demonstrated that the solution quality of such algorithms is significantly lower compared to that of shrinking algorithms, which iteratively simplify the problem. Consequently, this type of algorithm is not further explored in this work.

### 3.3.3 Update Rules

The update rule for one-point correlations is straightforward; defining an effective update rule for two-point correlations is more complex. Here, we propose approaches that work, but with a lower solution quality compared to our rules.

In the case of positive two-point correlations, we explored merging correlated items, because adding one item increases the likelihood that including the other also contributes to the optimal solution. However, we observed that the newly created merged item often had a lower one-point correlation than its individual components. This reduction in correlation sometimes caused other items to be prioritized in the selection process, potentially resulting in a full knapsack before the merged item could be added (leading to non-optimality).

For negative two-point correlations, we initially attempted to remove both correlated items in a single iteration. However, we found that it is generally preferable to add more items to the solution rather than deleting many items simultaneously. Excessive deletion can lead to situations where the knapsack still has available capacity, but no items remain in the problem for selection. Additionally, correlations tend to become more accurate with each iteration, particularly enhancing the reliability of one-point correlations. Consequently, we chose to delete items only when there is confidence in their irrelevance to the optimal solution.

# 4 Results

This chapter presents the results of numerical simulations comparing the performance of quantum algorithms (QIRO, MinQ, MMQ, and MaxQ) against classical heuristics (Greedy and Advanced Greedy) for the KP. To validate solution quality, we use DP to compute the optimal solution, leveraging its guaranteed optimality based on the optimal substructure property of the KP [2].

For additional context, other potential comparison techniques include approximation algorithms and schemes, which offer alternative solution strategies for the KP [20].

We first benchmark general cases that are randomly created within a given problem size. Those general cases aim to represent simple scenarios. Then, we analyze special cases in that the Greedy algorithm fails to find the optimal solution. In those special cases, we aim to show that our algorithms return a similar solution quality for low depth $p = 1$, but return a higher solution quality for higher depths $1 < p \leq 3$. This is of our interest because it might show that the algorithms can make more accurate decisions by leveraging quantum information.

All simulations were executed on a MacBook Air with an M1 chip, 8 GB of RAM, and macOS Sequoia (15.1).

## 4.1 Required Number of Qubits for the Knapsack Problem

To ensure compliance with the weight constraint, the KP formulation uses slack variables. Each slack variable corresponds to a distinct weight in the range $1, \ldots, W$, requiring one qubit per slack variable, as indicated in Equations 2.1 and 2.2. Besides, every item requires one qubit indicating if it is part of the solution. Consequently, the total number of required qubits is $W + N$. Our analysis focuses on small problem instances with up to 24 qubits because we use quantum simulation on classical hardware which is memory-intensive.

## 4.2 Benchmarking Procedure

Our benchmarking process first obtains the optimal solution through DP, then runs each algorithm to assess approximation quality. For each case, the parameter $A$ is

fine-tuned to enhance solution quality and $B = 1$ for simplicity. If the returned solution of one depth $p$ is not optimal, we run our algorithms for higher depths $p$. We define the approximation ratio $AR$ in percent as follows:

$$AR = \frac{V_i}{V_{DP}} * 100$$

where $V_{DP}$ represents the optimal solution value from DP and $V_i$ denotes the value returned by algorithm $i \in \{\text{QIRO, MinQ, MMQ, MaxQ}\}$.

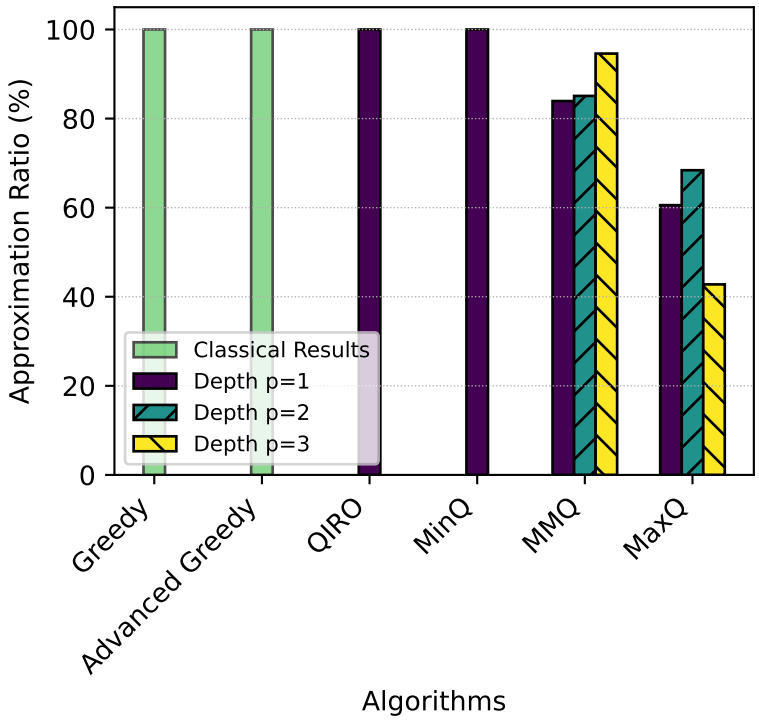
## 4.3 General Cases

This section introduces our general test cases and presents their benchmarking results. The following cases aim to construct simple scenarios. We try to show that leveraging quantum information might lead to higher solution quality in comparison to simple heuristics like the Greedy algorithm. To align with classical hardware constraints for the quantum simulation, we limit the maximum number of qubits to 24. Item weights are randomly selected within the range $\{1, \ldots, W\}$.

### 4.3.1 Small Case with 12 Required Qubits

In this test case, we examine four small KP instances where $N + W = 12$:

| Case | Number of Items ($N$) | Maximum Weight ($W$) | Value Assignment |
|---|---|---|---|
| 1 | 2 | 10 | Weight equals value |
| 2 | 2 | 10 | Value random (0–200) |
| 3 | 3 | 9 | Weight equals value |
| 4 | 3 | 9 | Value random (0–200) |



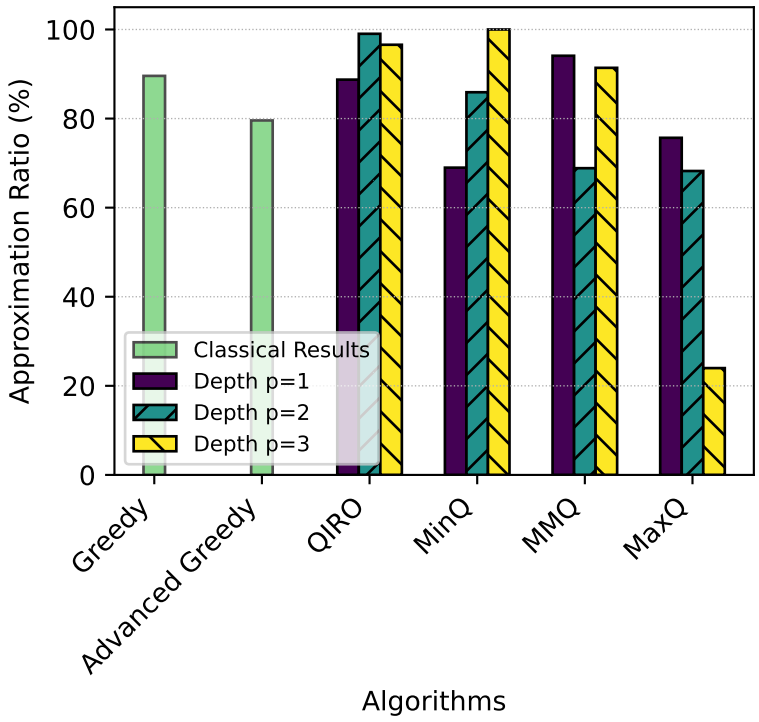

| Algorithm | Depth 1 | Depth 2 | Depth 3 |
|---|---|---|---|
| Greedy | 100 | | |
| Advanced | 100 | | |
| QIRO | 100 | | |
| MinQ | 100 | | |
| MMQ | 83.93 | 85.08 | 94.59 |
| MaxQ | 60.56 | 68.40 | 42.77 |

Figure 4.1: Approximation ratios in percent for small-sized cases with $A = \max(c_\alpha)$ for QIRO, MinQ and MaxQ, $A = \max(c_\alpha) + 1$ for MMQ. The table lists the approximation ratios in percent, with "Depth 1" also representing classical results for simplicity.

As shown in Figure 4.1, QIRO and MinQ achieve optimal solutions at low depth $p = 1$. For MMQ, the solution quality improves slightly at higher depths, as we expect due to quantum correlations that consider more information about the problem for higher depths $p$. However, MaxQ does not show improvement at increased depths, its returned value even decreases, which is unexpected behavior. Because of this unexpected behavior, we investigated MaxQ with different values for the fine-tuning parameters $A$ but we were not able to find a significantly improved solution. We do not understand the behavior of MaxQ fully yet and it needs further investigation.

### 4.3.2 Medium Case with 16 Required Qubits

For medium instances where $N + W = 16$, we use the following configurations:




| Algorithm | Depth 1 | Depth 2 | Depth 3 |
|---|---|---|---|
| Greedy | 89.58 | | |
| Advanced | 79.58 | | |
| QIRO | 88.74 | 99.03 | 96.57 |
| MinQ | 68.98 | 85.91 | 100.00 |
| MMQ | 94.10 | 68.86 | 91.39 |
| MaxQ | 75.68 | 68.23 | 23.98 |

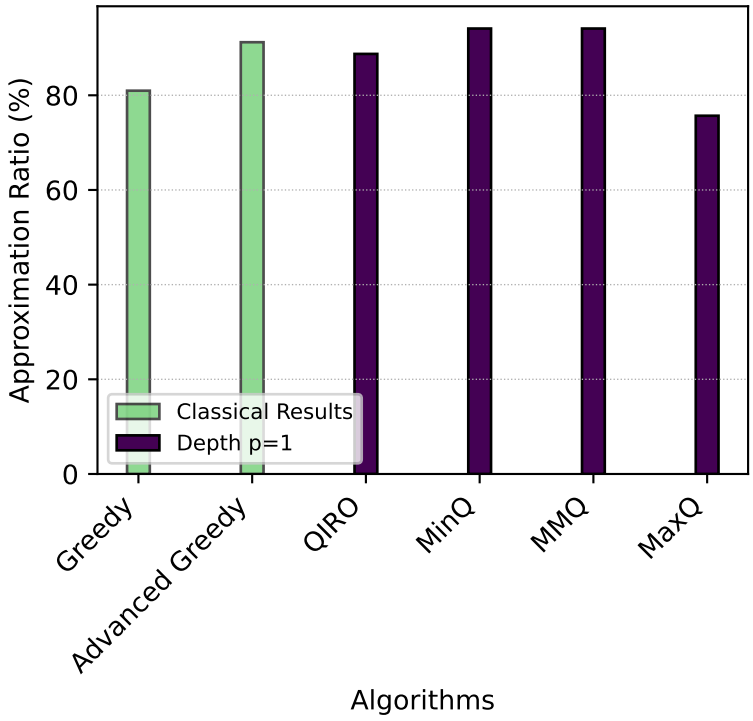
Figure 4.2: Approximation ratios in percent for medium-sized cases with $A = \max(c_\alpha)$ for QIRO, MMQ and MaxQ and $A = \max(c_\alpha) + 1$ for MinQ algorithms. The table lists the approximation ratios in percent, with "Depth 1" also representing classical results for simplicity.

| **Case** | **Number of Items ($N$)** | **Maximum Weight ($W$)** | **Value Assignment** |
|---|---|---|---|
| 1 | 4 | 12 | Weight equals value |
| 2 | 4 | 12 | Value random (0–200) |
| 3 | 6 | 10 | Weight equals value |
| 4 | 6 | 10 | Value random (0–200) |

In Figure 4.2 we see, that for medium cases, the solution qualities for QIRO and MinQ improve at higher depths $p$, while other algorithms show stable or reduced performance at higher depths. Notably, QIRO, MinQ, and MMQ outperform the Advanced Greedy algorithm. The increasing values that MinQ returns for increasing depths is how we expect the algorithms to work, as they gain more accurate correlations for higher depths $p$ and thus can make higher quality decisions. Interestingly, the solution quality of MMQ roughly stays constant for higher depths $p$ and the solution quality of MaxQ

even decreases (as in 4.3.1) for higher depths $p$. We explored the approximation ratio for MaxQ using different values for the fine-tuning parameter $A$ but the decreasing performance pattern stayed similar. Again, the behavior of MaxQ is weird and yet, we do not fully understand it.

### 4.3.3 Large Case with 24 Required Qubits




| Algorithm | Depth 1 |
|---|---|
| Greedy | 80.97 |
| Advanced | 91.21 |
| QIRO | 88.74 |
| MinQ | 94.10 |
| MMQ | 94.10 |
| MaxQ | 75.68 |

Figure 4.3: Approximation ratios in percent for large cases with depth $p = 1$ and $A = \max(c_\alpha) + 1$. The table lists the approximation ratios in percent, with "Depth 1" also representing classical results for simplicity.

In large cases with $N + W = 24$, we analyze the following instances:

| **Case** | **Number of Items ($N$)** | **Maximum Weight ($W$)** | **Value Assignment** |
|---|---|---|---|
| 1 | 8 | 16 | Weight equals value |
| 2 | 8 | 16 | Value random (0–200) |
| 3 | 10 | 14 | Weight equals value |
| 4 | 10 | 14 | Value random (0–200) |

Due to the computational intensity of simulating 24 qubits, we limit the depth exploration to $p = 1$, as simulating larger depths demands exponentially greater resources [22]. As shown in Figure 4.3, our algorithms, except MaxQ, return solutions with higher approximation ratios than the greedy algorithm at low depth $p = 1$. Most interestingly, we see that MinQ and MMQ return a slightly higher value than Advanced Greedy. QIRO returns a value in between Greedy and Advanced Greedy. MaxQ returns a lower total value than all other algorithms. This case indicates, that QIRO, MinQ and MMQ might be competitive to traditional heuristics for small problem instances regarding their returned value.

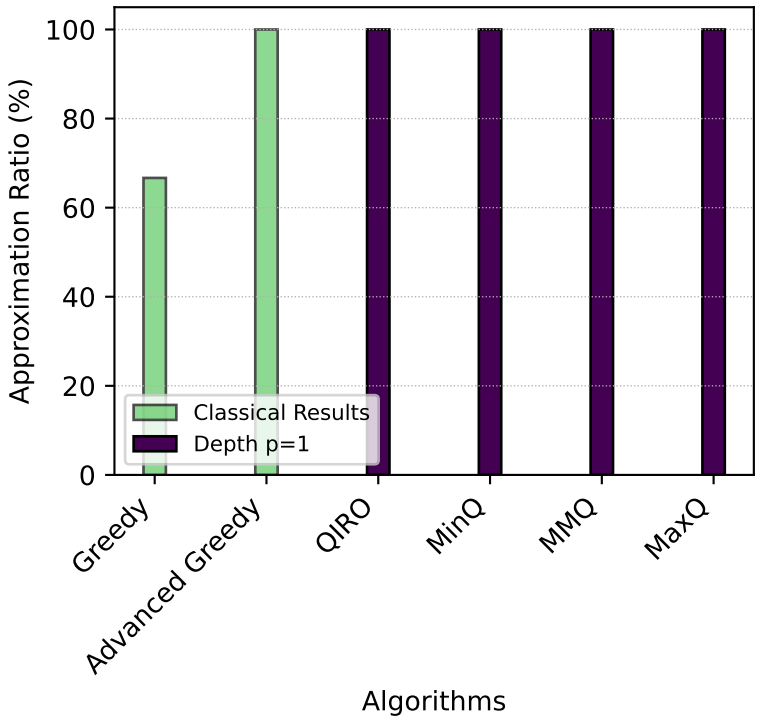
## 4.4 Special Cases

In this section, we present benchmarks on special cases that the Greedy algorithm fails to solve to optimality. The goal of our algorithms in those special cases is to return at least the same solution quality as the Greedy algorithm for low depth $p = 1$ but a higher solution qualtiy for higher depths $1 < p \leq 3$. Those cases are important because they provide us with insights if our proposed algorithms can leverage added quantum information to improve upon the solution quality of classical heuristics in challenging cases.

### 4.4.1 High Value-to-Weight Ratio Leading to Suboptimality

In this scenario, the weight constraint is $W = 9$, with $N = 3$ items requiring $N + W = 12$ qubits. We use the following items:

| **Item** | **Weight ($w$)** | **Value ($c$)** | **Value-to-Weight Ratio ($c/w$)** |
|---|---|---|---|
| 1 | 6 | 30 | 5 |
| 2 | 2 | 14 | 7 |
| 3 | 3 | 18 | 6 |



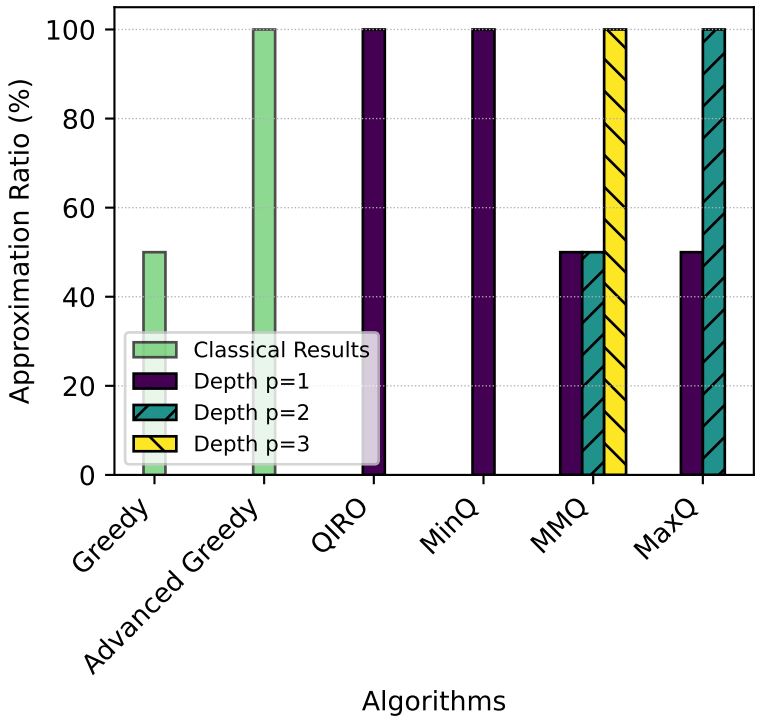

| Algorithm | Depth 1 |
|---|---|
| Greedy | 66.67 |
| Advanced | 100 |
| QIRO | 100 |
| MinQ | 100 |
| MMQ | 100 |
| MaxQ | 100 |

Figure 4.4: Approximation ratios in percent for the *high value-to-weight ratio leading to suboptimality* case with $A = \max(c_\alpha) + 4$ for QIRO, MinQ and MMQ and $A = \max(c_\alpha) + 2$ for MaxQ. The table lists the approximation ratios in percent, with "Depth 1" also representing classical results for simplicity.

The Greedy algorithm selects items 2 and 3, resulting in a suboptimal total value of 32. However, the optimal solution consists of items 1 and 3, achieving a total value of 48. The failure of the Greedy algorithm is due to its reliance on value-to-weight ratios, which prioritize smaller, high-ratio items over larger, higher-value items. As shown in Figure 4.4, QIRO, MinQ, MMQ and MaxQ identify the optimal solution at low depth $p = 1$. Showcasing, that our algorithms can find the optimal solution in this confusing value-to-weight ratio scenario for the greedy algorithm using low depth $p = 1$.

### 4.4.2 High-Value, Large-Weight Items

For this case, we set $W = 10$ with $N = 2$ items, requiring $N + W = 12$ qubits:



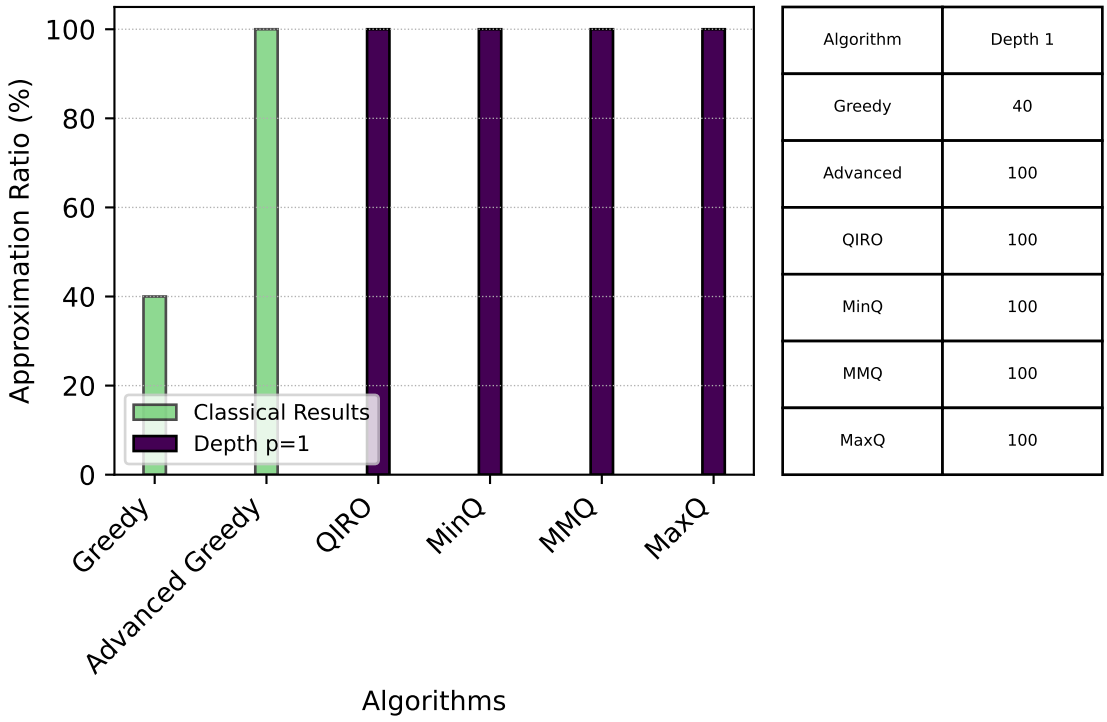

| Algorithm | Depth 1 | Depth 2 | Depth 3 |
|---|---|---|---|
| Greedy | 50 | | |
| Advanced | 100 | | |
| QIRO | 100 | | |
| MinQ | 100 | | |
| MMQ | 50 | 50 | 100 |
| MaxQ | 50 | 100 | |

Figure 4.5: Approximation ratios in percent for the *high-value, large-weight items* case with $A = \max(c_\alpha) + 3$ for all algorithms. The table lists the approximation ratios in percent, with "Depth 1" also representing classical results for simplicity.

| **Item** | **Weight ($w$)** | **Value ($c$)** | **Value-to-Weight Ratio ($c/w$)** |
|---|---|---|---|
| 1 | 9 | 100 | 11.1 |
| 2 | 2 | 50 | 25 |

The optimal solution includes item 1, which provides the highest total value. However, its large weight and lower value-to-weight ratio lead the Greedy algorithm to select item 2, resulting in a suboptimal solution. At low depth $p = 1$, QIRO and MinQ find the optimal solution by selecting item 2. As the depth $p$ increases, MMQ and MaxQ are also able to find the optimal solution. This improvement demonstrates that MMQ and MaxQ in this case are able to improve their decisions using more quantum information. Finding the optimal solutions using increasing depths $p$ is how our algorithms ideally work.

### 4.4.3 Competing Small and Large High-Value Items




| Algorithm | Depth 1 |
|---|---|
| Greedy | 40 |
| Advanced | 100 |
| QIRO | 100 |
| MinQ | 100 |
| MMQ | 100 |
| MaxQ | 100 |

Figure 4.6: Approximation ratios in percent for the *competing small and large high-value items* case with $A = \max(c_\alpha)$ for QIRO, MinQ and MMQ and $A = \max(c_\alpha) + 3$ for MaxQ. The table lists the approximation ratios in percent, with "Depth 1" also representing classical results for simplicity.

Here, the weight constraint is $W = 9$ with $N = 3$ items, requiring $N + W = 12$ qubits:

| **Item** | **Weight ($w$)** | **Value ($c$)** | **Value-to-Weight Ratio ($c/w$)** |
|---|---|---|---|
| 1 | 8 | 100 | 11.1 |
| 2 | 1 | 20 | 20 |
| 3 | 2 | 30 | 15 |

The optimal solution consists of item 1, which has the highest total value. However, the Greedy algorithm selects items 2 and 3 due to their higher value-to-weight ratios, resulting in a suboptimal solution. As shown in Figure 4.6, all quantum algorithms successfully identify the optimal solution at low depth $p = 1$. This result shows the

ability of our algorithms to outperform Greedy in this case at low depths $p$ without leveraging quantum information.

# 5 Discussion

Reflecting on the results of our benchmarking, we observe that the solution quality of our algorithms QIRO and MinQ generally improves with increasing depth $p$. However, for MMQ, our results do not generally indicate improved behavior for higher depths $p$ as it stagnates in the general medium case (as observed in section 4.3.2). For MaxQ, we observed weird behavior in general cases, as the approximation ratios often decreases for higher depths $p$. Yet, we do not fully understand this behavior and it requires further investigation. While the performances in the general cases are sometimes weird, the performances for special cases suggest expected behavior and our algorithms are able to find the optimal solution for every case and for increasing depths $p$. Due to this observation, we expect the application of our algorithms to be niche and tailored to special cases.

One significant factor contributing to the returned solutions of our algorithms lies in the update rules. While our rules are designed to generalize across KP instances, they may not be equally effective for all problem types. Individualized update rules tailored to specific problem structures or instances could potentially enhance performance.

Hardware limitations also played a role in constraining our experiments. Due to the current error-prone nature of quantum devices, we relied exclusively on quantum simulations, which, in our case, were limited to 8 GB of RAM. This constraint likely impacted both the execution time and the scalability of our simulations, as quantum simulations are memory-intensive. With access to larger memory resources, we anticipate that simulation times could be significantly reduced, enabling the exploration of more complex problem instances and higher-depth circuits. As a future work, it would be interesting to run our algorithms on real quantum devices that are not constrained by memory-size and compare those results to our results.

This work opens several avenues for future research. For instance, extending MinQ and MaxQ to incorporate two-point correlations could improve execution efficiency by allowing the simultaneous inclusion or exclusion of multiple items per iteration. This modification could accelerate convergence while maintaining or even improving solution quality. Furthermore, a deeper investigation into fine-tuning could reveal patterns for specific problem categories, enabling the development of adaptive tuning strategies that optimize performance across a wider range of problem instances.

Another promising extension involves integrating classical heuristics with quantum

algorithms through warm-starting techniques. By leveraging classical methods to generate an initial solution, the quantum algorithm could begin with a meaningful starting point, potentially enhancing its ability to converge on superior solutions. This hybrid approach could combine the strengths of classical and quantum optimization, offering improved performance.

Different methods of calculating the quantum correlations are also a matter of future exploration. For instance, Finžgar *et al.* [23] propose a new method to design quantum annealing schedules based on Bayesian optimization. This and other approaches could improve our work in regard to solution quality.

# 6 Conclusion

This thesis explored the application of iterative quantum optimization algorithms for solving the KP. By leveraging quantum computing paradigms, specifically the QAOA and its iterative extensions, we developed and benchmarked algorithms capable of incorporating quantum correlations to iteratively simplify and solve the KP.

The results demonstrate that the solution quality of the proposed quantum algorithms, including QIRO, MinQ, MMQ, and MaxQ, generally improves with increasing circuit depth $p$ in special cases. However, in general cases, the performance of MMQ suggests that its returned value stagnates in some cases for increasing depth $p$. For MaxQ, the solution quality decreases for higher depths $p$ in general cases, which is unexpected behavior and requires further investigation. The performance of QIRO and MinQ align with the theoretical premise that higher depths enable the capture of more complex problem-specific correlations in both special and general cases. We expect our algorithms to have applications in niche cases that classical heuristics fail to solve to optimality.

The study underlined the critical role of update rules informed by one- and two-point quantum correlations. While these rules performed effectively in guiding item inclusion and exclusion, the results suggest opportunities for improvement, particularly in addressing negative correlations and multi-item interactions, which could further enhance algorithm robustness. The algorithms effectively addressed specifically designed toy scenarios where the Greedy algorithm failed to find the optimal solution. Fine-tuning parameterization emerged as a critical factor in balancing solution quality and computational efficiency, marking further investigation into systematic fine-tuning methodologies.

This work extends the scope of quantum optimization by introducing and benchmarking iterative quantum algorithms tailored to the KP. By demonstrating the feasibility of leveraging quantum correlations to simplify complex optimization problems, it provides a foundation for future research into hybrid quantum-classical optimization strategies.

# List of Figures

# Bibliography


[1] E. Kaya, B. Gorkemli, B. Akay, and D. Karaboga, *A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems*, 2022. DOI: https://doi.org/10.1016/j.engappai.2022.105311.

[2] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004, pp. 20–26.

[3] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004, pp. 27–29.

[4] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm*, 2014. arXiv: 1411.4028 [quant-ph].

[5] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, *Obstacles to variational quantum optimization from symmetry protection*, Dec. 2020. DOI: 10.1103/PhysRevLett.125.260505.

[6] J. R. Finžgar, A. Kerschbaumer, M. J. Schuetz, C. B. Mendl, and H. G. Katzgraber, *Quantum-informed recursive optimization algorithms*, May 2024. DOI: 10.1103/PRXQuantum.5.020327.

[7] L. T. Brady and S. Hadfield, *Iterative quantum algorithms for maximum independent set*, Nov. 2024. DOI: 10.1103/PhysRevA.110.052435.

[8] W. Bożejko, A. Burduk, J. Pempera, *et al.*, *Optimal solving of a binary knapsack problem on a d-wave quantum machine and its implementation in production systems*, Published online, awaiting volume and page numbers, 2024. DOI: 10.1007/s10479-024-06025-1.

[9] W. van Dam, K. Eldefrawy, N. Genise, and N. Parham, *Quantum optimization heuristics with an application to knapsack problems*, 2022. arXiv: 2108.08805 [quant-ph].

[10] A. Awasthi, F. Bär, J. Doetsch, H. Ehm, M. Erdmann, M. Hess, J. Klepsch, P. A. Limacher, A. Luckow, C. Niedermeier, L. Palackal, R. Pfeiffer, P. Ross, H. Safi, J. Schönmeier-Kromer, O. von Sicard, Y. Wenger, K. Wintersperger, and S. Yarkoni, *Quantum computing techniques for multi-knapsack problems*, 2023. DOI: 10.1007/978-3-031-37963-5_19.

[11] S. Wilkening, A.-I. Lefterovici, L. Binkowski, M. Perk, S. Fekete, and T. J. Osborne, *A quantum algorithm for the solution of the 0-1 knapsack problem*, 2023. arXiv: 2310.06623 [quant-ph].

[12] F. Glover, G. Kochenberger, and Y. Du, *A Tutorial on Formulating and Using QUBO Models*, Nov. 2018. DOI: 10.48550/arXiv.1811.11538. arXiv: 1811.11538 [cs.DS].

[13] A. Lucas, *Ising formulations of many np problems*, 2014. DOI: 10.3389/fphy.2014.00005.

[14] A. Cervera-Lierta, *Exact Ising model simulation on a quantum computer*, Dec. 2018. DOI: 10.22331/q-2018-12-21-114.

[15] M. Assi and R. A. Haraty, *A survey of the knapsack problem*, 2018. DOI: 10.1109/ACIT.2018.8672677.

[16] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004, pp. 1–5.

[17] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004, pp. 15–16.

[18] F. Glover, *Advanced greedy algorithms and surrogate constraint methods for linear and quadratic knapsack and covering problems*, 2013. DOI: 10.1016/j.ejor.2013.04.010.

[19] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004, pp. 11–14.

[20] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004, pp. 29–42.

[21] V. Fischer, M. Passek, F. Wagner, J. R. Finžgar, L. Palackal, and C. B. Mendl, *The role of quantum and classical correlations in shrinking algorithms for optimization*, 2024. arXiv: 2404.17242 [quant-ph].

[22] Y. Zhou, E. M. Stoudenmire, and X. Waintal, *What limits the simulation of quantum computers?* Nov. 2020. DOI: 10.1103/PhysRevX.10.041038.

[23] J. R. Finžgar, M. J. A. Schuetz, J. K. Brubaker, H. Nishimori, and H. G. Katzgraber, *Designing quantum annealing schedules using bayesian optimization*, Apr. 2024. DOI: 10.1103/PhysRevResearch.6.023063.