# Software aided approach for constrained optimization based on QAOA modifications

Tomasz Lamża[1,2], Justyna Zawalska[1,2][0000−0003−3189−7618], Mariusz Sterzel[2] and Katarzyna Rycerz[1,2][0000−0002−8032−7251]

[1] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Krakow, Poland
[2] Academic Computer Center Cyfronet AGH, ul. Nawojki 11, 30-950 Krakow, Poland
`tomasz.lamza@cyfronet.pl, justyna.zawalska@cyfronet.pl, kzajac@agh.edu.pl`

**Abstract.** We present two variants of the QAOA modification for solving constrained combinatorial problems. The results presented in this paper were obtained using the QHyper framework, which we developed specifically for this purpose. More specifically, we use the created framework to compare the QAOA results with its two modifications, namely: Weight-Free QAOA (WF-QAOA) and Hyper QAOA (H-QAOA). Additionally, we compare the Basin-hopping global optimization method for subsequent sampling of the initial points for the proposed QAOA modifications with a simple Random Search. The results obtained for the Knapsack Problem indicate that the proposed solution outperforms the original QAOA algorithm and can be promising for QUBO, where adjusting the relative importance of the cost function and the constraints is a significant challenge.

**Keywords:** QAOA · Constrained Optimization · Hyperparameters · QUBO · Penalty

## 1 Introduction

Recently, using and developing quantum algorithms for solving combinatorial problems is becoming a popular subject in the research field of quantum computation. Examples of the main findings in this area include variational algorithms for gate-based quantum devices, such as the Quantum Approximate Optimization Algorithm (QAOA) [4], or quantum annealing solvers realized by D-Wave devices[3]. However, successful usage of such algorithms usually requires careful setting of their initial parameters, which is quite crucial yet nontrivial task. Furthermore, the required formulation of the objective function as Quantum Unconstrained Binary Optimization (QUBO) in most cases requires cautious setting of weights between the cost function and the constraints. This is also not obvious; therefore, the weights often become additional hyperparameters of the problem. There are also attempts to efficiently search for such parameters and hyperparameters [1, 13, 15] but research in this direction is in its early

[3] `https://www.dwavesys.com/`

development stage. In this paper, we propose two variants of the QAOA modification that improve optimization for constrained problems. The first modification (Weight-Free QAOA or WF-QAOA) is based on the definition of the new weight-free observable for the QAOA-based variational algorithm. Furthermore, this allows extending the variational parameter set with the QUBO weights in the second modification (Hyper QAOA or H-QAOA). When comparing the performance of the proposed variants, we examined two ways of sampling the initial points for the variational algorithms: simple Random Search and global optimizer Basin-hopping [18]. Additionally, to facilitate conducting a variety of planned experiments, we introduce the architecture of the QHyper software framework as a modular tool for researchers. The results show that the proposed variants perform better than the regular QAOA algorithm. Random Search with WF-QAOA produced the best results when the number of variational algorithm launches was small. However, Basin Hopping variants performed better with a larger number of the algorithm launches, particularly when using multiple initial points. Although we present the results using the Knapsack Problem, the proposed solution can be used for any combinatorial problem transformed to QUBO.

This paper is organized as follows: in Section 2 we introduce important preliminaries and in Section 3 we describe related work. The proposed variants of the QAOA modification are shown in Section 4. The QHyper experiment framework is presented in Section 5. Section 6 discusses different sampling methods based on global optimizers used in our experiments. The results are presented in Section 7 and conclusions can be found in Section 8.

## 2 Preliminaries

In this section, we provide a concise overview of the key aspects related to quantum solutions for combinatorial problems necessary to enhance the the readability of the paper.

### 2.1 Combinatorial Problem Formulation Models

In the Constrained Quadratic Model[4] (CQM), the cost function $C_f(\boldsymbol{x})$ can be represented using an $n \times n$ cost matrix $\boldsymbol{C}$ where

$$C_f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x} = \sum_i c_{ii} x_i + \sum_{i<j} c_{ij} x_i x_j + const. \tag{1}$$

Similarly, the constraints are given by functions $G_f^{(k)}(\boldsymbol{x})$ represented by an $n \times n$ cost matrix $\boldsymbol{G}^{(k)}$ where

$$G_f^{(k)}(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{G}^{(k)} \boldsymbol{x} = \sum_i g_{ii}^{(k)} x_i + \sum_{i<j} g_{ij}^{(k)} x_i x_j + const \circ 0 \tag{2}$$

[4] `https://docs.ocean.dwavesys.com/en/stable/concepts/cqm.html`

where $k = 1, \dots, M$ and $M$ is the number of constraints. The variables $\{x_i\}_{i=1,\dots,N}$ can be binary or integer and the matrices $\boldsymbol{C}$ , $\boldsymbol{G}^{(k)}$ are real-valued. The symbol $\circ$ denotes a comparison operator $\{\geq, \leq, =\}$.

To use quantum solvers, the CQM problem must be transformed into the forms of (1) and (2), where all $\{x_i\}_{i=1,\dots,N}$ are binary. Additionally, all functions have to be combined into Quantum Unconstrained Binary Optimization. In the first step, this is done by transforming the constraints $G_f^{(k)}(\boldsymbol{x}) \circ 0$ into the equality constraints $K_f^{(k)}(\boldsymbol{x}) = 0$ where $\forall \boldsymbol{x}, K_f^{(k)}(\boldsymbol{x}) \geq 0$. Next, by adding weighted constraints and the cost function together, the objective function is obtained in the form

$$f_{QUBO}(\boldsymbol{x}) = \alpha_0 C_f(\boldsymbol{x}) + \sum_{k=1}^{M} \alpha_k K_f^{(k)}(\boldsymbol{x}), \tag{3}$$

where $\alpha_0$ is the weight of the cost function and $\alpha_k$ is the weight of the $k$-th constraint, $\alpha_i > 0$.

## 2.2 Quantum Approximate Optimization Algorithm

The QAOA is a variational combinatorial optimization algorithm for gate-based quantum devices. To apply the algorithm, the objective function $f_{QUBO}(\boldsymbol{x})$ is translated into the cost Hamiltonian $H_C$

$$H_C = \alpha_0 H_{C_f} + \sum_{k=1}^{M} \alpha_k H_{K_f^{(k)}}, \tag{4}$$

where $H_{C_f}$ and $H_{K_f^{(k)}}$ are the Hamiltonians that encode the cost function and the constraints, respectively. In addition, a mixing Hamiltonian $H_M$ that is not commuting with $H_C$ is required. A common choice is $H_M = \sum_{i=1}^{N} X_i$, where $X_i$ is the Pauli-X gate. The QAOA ansatz consists of $p$ alternating layers $U_C$ and $U_M$

$$U_C(\gamma) = e^{-i\gamma H_C}, \tag{5}$$

$$U_M(\beta) = e^{-i\beta H_M}, \tag{6}$$

where $\gamma, \beta$ are adjustable parameters. The algorithm uses a chosen classical optimizer to minimize the expectation value

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta} | \, H_C \, | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle \,, \tag{7}$$

of the quantum state prepared with a quantum device (or its simulator):

$$| \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle = U_M(\beta_p) U_C(\gamma_p) \cdots U_M(\beta_1) U_C(\gamma_1) \, |+\rangle^{\otimes n} \,, \tag{8}$$

where $|+\rangle^{\otimes n}$ is a uniform superposition of $n$ qubits. The optimization is performed with respect to $2p$ parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta}) \in [0, 2\pi]^p \times [0, \pi]^p$.

### 2.3 Knapsack Problem

The Knapsack Problem is a combinatorial optimization problem where given a set of items, each of a certain weight and value, the aim is to determine which items should be packed to the knapsack to maximize the total value of selected items while not exceeding the knapsack's weight limit. The problem can be represented as a QUBO [9]

$$f_{QUBO\_KP}(\boldsymbol{x},\boldsymbol{y}) = -\alpha_0 \sum_{i=1}^{N} c_i x_i + \alpha_1 (1 - \sum_{i=1}^{W} y_i)^2 + \alpha_1 (\sum_{i=1}^{W} i y_i - \sum_{i=1}^{N} w_i x_i)^2, \quad (9)$$

where $N$ is the number of items available, $W$ is the maximum weight of the knapsack, $c_i$ and $w_i$ are the value and weight of the item $i$, respectively. $\boldsymbol{x} = [x_i]_N$ is a Boolean vector, where $x_i = 1$ if and only if the item $i$ was selected to be inserted into the knapsack. $\boldsymbol{y} = [y_i]_W$ is a one-hot vector where $y_i = 1$ if and only if the weight of the knapsack is equal to $i$. In this paper, we focus on the version from [9], where the weight for both of the constraints is the same. That is, $\alpha_0$ is the weight of the cost function and $\alpha_1$ is the weight of the constraints.

## 3 Related Work

The original version of the QAOA [4] focuses on solving unconstrained binary optimization problems. However, constrained problems can also be solved by this algorithm. There are two main methods to incorporate constraints into the QAOA [8]. The first of them, called the *hard* constraints, is based on designing a Quantum Approximate Optimization Ansatz with a special mixing Hamiltonian that prevents finding unfeasible solutions [7]. Examples of such mixing Hamiltonians include $XY$ mixers (and their variations) [5, 8, 19], Grover mixers [2], and problem-specific mixers based on the use of quantum machine learning [14].

Another option — called *soft* constraints — is incorporating the constraints in the objective function (in the form of QUBO) and using the standard $X$ mixing Hamiltonian. In this case, setting optimal weights between the cost function and the constraints that will allow obtaining feasible results is a major challenge. There are several methods available to select appropriate weights. These include ways of static settings of weights using the information taken from the objective function, such as the maximum QUBO coefficient or the maximum absolute value of the difference achieved by flipping one QUBO variable [1]. There are also efforts of using Monte Carlo-based iterative improvement of distribution from which hyperparameters are sampled, such as the Cross Entropy Method [15] or Tree-structured Parzen Estimator [13]. Monte Carlo methods are also an inspiration for global optimizers such as SHGO [3] or Basin-hopping [18]. As these methods originated from effort towards finding the minimum energy structure for molecules [12], their usefulness in the context of Quantum Variational Algorithms is a promising approach [6, 10].

To the best of our knowledge, the QAOA modification proposed in this paper (see Section 4) was never considered in the literature. Additionally, although most of the libraries for gate-based quantum computation provide an implementation of variational algorithms such as QAOA and VQE [16], performing experiments that join global optimizers with QAOA variants requires additional tedious effort from the researchers. The presented QHyper system (see Section 5) is intended to fill this gap.

## 4 Proposed QAOA modifications for constrained problems

In this section, we propose two QAOA modifications for QUBO-based constrained problems. The obtained experiment results for the Knapsack Problem described in Section 7 indicated that presented variants seem to be a promising alternative for the QAOA.

### 4.1 Weight-free Hamiltonian

A constrained problem presented in the Hamiltonian form (see Eq. 4) as a diagonal matrix requires appropriate weights ($\boldsymbol{\alpha}$) setting so that the lower energy states are better solutions than higher energy ones. In this paper, we propose the formulation of an alternative weight-free diagonal Hamiltonian that satisfies the condition of appropriate order of energies. The new Hamiltonian is used for the expectation value estimation (see Eq. 7). The formulation is valid for problems with cost function satisfying $C_f(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x}$, where $\boldsymbol{x} = [x_i]_N$ is a binary vector that encodes the solution fulfilling constraints (see Eq. 3). This is true for numerous combinatorial problems including the Knapsack Problem. Under this assumption, we define the Hamiltonian $H_{wf}$ as a diagonal matrix of the order $2^N$, namely

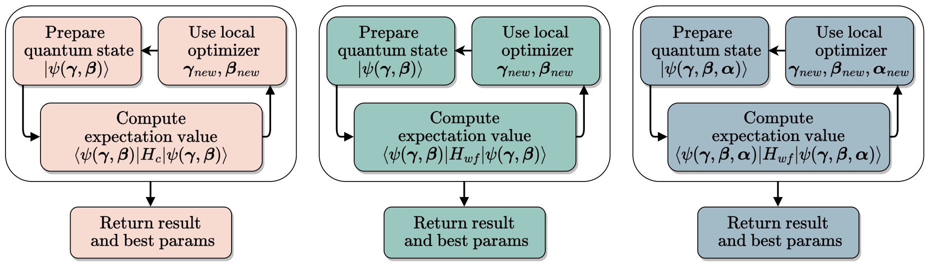
$$H_{wf}[\text{decimal}(\boldsymbol{x}), \text{decimal}(\boldsymbol{x})] = \begin{cases} -C_f(\boldsymbol{x}) & \text{if } \boldsymbol{x} \text{ satisfies constraints} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

**Algorithm 1** Estimating Expectation Value with Weight-Free Hamiltonian

```
Require: results - vector of binary results from anzatz measurements in shots,
    shots_num - number of shots
 1: for $i = 1 \to$ shots_num do
 2:     sol ← results[i]                      ▷ taking binary vector of solution $i$
 3:     if sol satisfies constraints then
 4:         score ← $-C_f(sol)$               ▷ equivalent to $\langle sol| H_{wf} |sol\rangle$
 5:     else
 6:         score ← 0                         ▷ equivalent to $\langle sol| H_{wf} |sol\rangle$
 7:     end if
 8:     sum+ = score
 9: end for
10: exp_value = $\frac{\text{sum}}{\text{shots\_num}}$    ▷ estimating expectation value of the measured state
11: return exp_value
```

The main idea of the QAOA modification is to use an anzatz based on its original Hamiltonian (see Eq. 4) while changing the operator for estimating the expectation value. This is illustrated in Fig. 1b in comparison to the regular QAOA in Fig. 1a. The proposition can also be seen as a variant of the VQE algorithm valid due to the variational principle [16].

Since the weight-free Hamiltonian is not used for the anzatz building, there is also no need to present the full formula of the new Hamiltonian as the Ising model. We require only a simple algorithmic way of calculating eigenstates for a fixed number of eigenvectors that were measured to estimate the expectation value. This is presented as Algorithm 1.


Prepare quantum state $|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle$
Use local optimizer $\boldsymbol{\gamma}_{new},\boldsymbol{\beta}_{new}$
Compute expectation value $\langle\psi(\boldsymbol{\gamma},\boldsymbol{\beta})|H_c|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle$
Return result and best params
Prepare quantum state $|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle$
Use local optimizer $\boldsymbol{\gamma}_{new},\boldsymbol{\beta}_{new}$
Compute expectation value $\langle\psi(\boldsymbol{\gamma},\boldsymbol{\beta})|H_{wf}|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle$
Return result and best params
Prepare quantum state $|\psi(\boldsymbol{\gamma},\boldsymbol{\beta},\boldsymbol{\alpha})\rangle$
Use local optimizer $\boldsymbol{\gamma}_{new},\boldsymbol{\beta}_{new},\boldsymbol{\alpha}_{new}$
Compute expectation value $\langle\psi(\boldsymbol{\gamma},\boldsymbol{\beta},\boldsymbol{\alpha})|H_{wf}|\psi(\boldsymbol{\gamma},\boldsymbol{\beta},\boldsymbol{\alpha})\rangle$
Return result and best params

(a) QAOA (b) WF-QAOA (c) H-QAOA

Fig. 1: Comparison of execution schemes for regular QAOA(a), Weight-free QAOA(b) and Hyper QAOA(c)

## 4.2 Alternate variational parameter set

For anzätze built with weighted Hamiltonians (see Eq. 4), the measurement result depends not only on the choice of the $\boldsymbol{\gamma}, \boldsymbol{\beta}$ angles but also on the Hamiltonian weights – hyperparameters $\boldsymbol{\alpha}$. Based on Eq. 4 and Eq. 5 it can be observed that the $U_C$ layer that encodes the objective function of the optimization problem is parameterized by both the angle $\gamma_l$ and the weights $\boldsymbol{\alpha}$

$$\begin{aligned} \forall l \in 1,\dots,p \quad \widehat{U_C}(\boldsymbol{\alpha},\gamma_l) &= exp(-i \cdot \gamma_l(\alpha_0 H_{C_f} + \sum_{k=1}^{M} \alpha_k H_{K_f^{(k)}})) \\ &= exp(-i \cdot 1(\gamma_l \alpha_0 H_{C_f} + \sum_{k=1}^{M} \gamma_l \alpha_k H_{K_f^{(k)}})) \\ &= \widehat{U_C}(\gamma_l \boldsymbol{\alpha}, 1). \end{aligned} \tag{11}$$

The vector of hyperparameters $\boldsymbol{\alpha}$ must be a proper multiplier for all of the $p$ angles $\boldsymbol{\gamma}$. As a result, $\boldsymbol{\alpha}$ can be treated as an additional variational parameter together with $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. Therefore, we propose the Hyper QAOA (H-QAOA) extension using such parameters set with a weight-free Hamiltonian as an observable (see Fig. 1c).

## 5 QHyper Experiment Framework

The experiments results presented in this paper were performed using the QHyper framework designed as a software support tool for researchers working with the problem of setting hyperparameters. The architecture of the system is shown in Fig. 2. Below, we present the main components of the system.

**Supported combinatorial optimization problems**. QHyper allows defining various optimization problems that can be solved by integrated solvers. The problem definition should include a cost function and a constraint list in the form of Sympy expressions[5]. The definitions should satisfy the requirements of the CQM model (see Section 2.1). QHyper supports the transformation of CQM to QUBO based on the D-Wave dimod library[6]. The produced QUBO can then be passed to any available solver in our framework. Currently, the library's limitations restrict the transformation of CQMs to QUBOs only to linear constraints. Inspired by the results from [15] we chose the Knapsack Problem as the use case for our research. However, the system includes other predefined problems such as multistage calculations planning (the Workflow Scheduling Problem [17]) or the Traveling Salesman Problem [9]. Custom problem definitions can also be added.

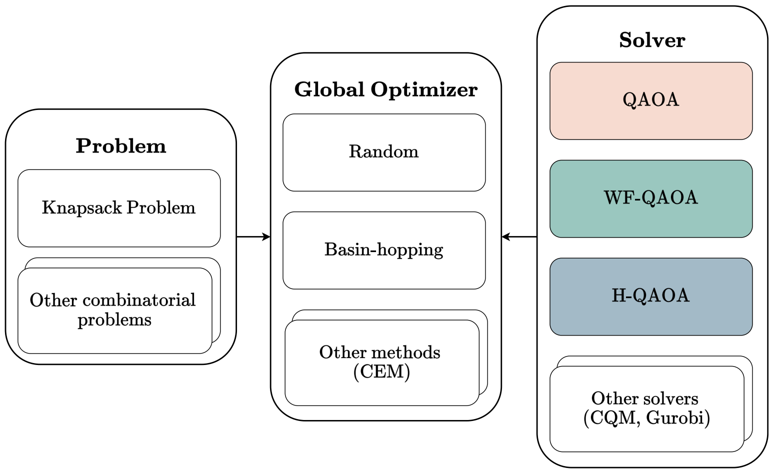


Fig. 2: QHyper architecture.

[5] `https://docs.sympy.org/latest/tutorials/intro-tutorial/intro.html`
[6] `https://docs.ocean.dwavesys.com/en/stable/docs_dimod/reference/generated/dimod.cqm_to_bqm.html`

**Supported solvers**. Currently, as shown in Fig. 2, the system is integrated with different types of solvers. Apart from the regular QAOA solver (see Section 2.2), modifications proposed in Section 4 are provided, namely: WF-QAOA described in Section 4.1 and H-QAOA with both modifications from Section 4. The solvers are implemented using the PennyLane[7] library. Additionally, the system also provides an overlay to the D-Wave Leap CQM Solver[8]. Another integrated solver is the purely classical solution Gurobi[9] that is used as a convenient reference method. The modular structure of the system enables effortless integration of additional solvers.

**Global optimizers.** We added the possibility of different sampling of the initial parameters set by using a predefined set of global optimizers. Currently, QHyper supports the following: Basin-hopping, the Cross Entropy Method (CEM), and the simple parallel optimizer based on a Random Search. The details of the possible variants used in this paper are presented in Section 6

```
# 1. Creating the Knapsack Problem instance
knapsack = KnapsackProblem(max_weight=1, items=[(1, 2), (1, 1)])
# 2. Specyfing the solver configuration
solver_config = {
    # Choosing the type of the local optimizer
    'optimizer': {
        'type': 'scipy',
        'maxfun': 200
    },
    # Choosing the type of the parameterized quantum circuit
    'pqc': {
        'type': 'hqaoa',
        'layers': 5,
    },
}
# 3. Specyfing the initial parameters configuration
params_config = {
    'angles': [[0.5]*5, [1]*5],
    'hyper_args': [1, 2.5, 2.5],
}
# 4. Creating the variational algorithm (VQA)
vqa = VQA(knapsack, config=solver_config)
# 5. Creating the Random global optimizer
random = Random(processes=5, number_of_samples=100,
        bounds=[[1, 10], [1, 10], [1, 10]])
# 6. Running the solver with the Random hyperoptimizer
best_params = vqa.solve(params_cofing, random)}
```

Code Listing 1.1: Sample usage of QHyper for solving a Knapsack Problem

[7] https://pennylane.ai/
[8] https://docs.dwavesys.com/docs/latest/doc_leap_hybrid.html)
[9] https://www.gurobi.com/

**Example usage.** Code snippet in the Listing. 1.1 shows the sample usage of the QHyper API. In the example, the H-QAOA solver is used with the default optimizer from the Scipy library (Broyden–Fletcher–Goldfarb–Shanno — BFGS [11]). As the example of a global optimizer Random Search was chosen.

## 6 Global Optimizer Variants

In this work we used QHyper to compare three variants of QAOA-inspired solvers described in Section 4. We employed two different probabilistic global optimizers. In the first case, we used a simple Random Search (RS) approach, where an $N$-element set of parameter vectors is sampled from the uniform distribution. Next, for each vector, a separate variational algorithm is launched. In the QHyper system, in the case of simulation of the quantum algorithm on the classical HPC machine, this is done in parallel (parameter study approach). After the execution, the best result is chosen. The detailed pseudocode is shown as Algorithm 2.

**Algorithm 2** Random Search

```
Require: n - samples number, a - vector of lower bounds, b - vector of upper bounds
 1: $x_1, ..., x_n \sim \mathcal{U}_{[\boldsymbol{a},\boldsymbol{b}]}$
 2: $v_1, ..., v_n \leftarrow$ variational_algorithm($x_1$), ..., variational_algorithm($x_n$)    ▷ This step
    can be done in parallel
 3: return $(x_i, v_i)$, where $v_i = min(v_1, ..., v_n)$
```

**Algorithm 3** Basin-hopping

```
Require: $x_0$ - initial guess, niter - number of Basin-hopping iterations, stepsize -
    maximum step size for use in the random displacement, a - vector of lower bounds,
    b - vector of upper bound
 1: $v\_x_0 \leftarrow$ variational_algorithm($x_0$)
 2: for $i = 1 \rightarrow$ niter do
 3:     $x \leftarrow$ random_step($x_0$)        ▷ The step is chosen uniformly in the region from
    $x_0$-stepsize to $x_0$+stepsize, in each dimension
 4:     while $x \notin (\boldsymbol{a}, \boldsymbol{b})$ or $x$ doesn't meet all criteria do
 5:         $x \leftarrow$ random_step($x_0$)
 6:     end while
 7:     $v\_x \leftarrow$ variational_algorithm($x$)
 8:     if $v\_x < v\_x_0$ then
 9:         $x_0 \leftarrow x$
10:         $v\_x_0 \leftarrow v\_x$
11:     end if
12: end for
13: return $(x_0, v\_x_0)$
```

In the second case, single sampling with improvement was used on the example of the Basin-hopping algorithm [18]. A single parameter vector is sampled with the uniform distribution. Next, the variational algorithm is run and its output is accepted if the result is better. Finally, the next parameter vector is chosen uniformly in the region around the last accepted result. The process is repeated $N$ times and, after that, the best result is chosen. The detailed pseudocode of the Basin-hopping algorithm is shown as Algorithm 3.

## 7 Experiment Results

All experiments were performed on a Knapsack Problem instance with three items, each with a weight of 1 and values of 2, 2, and 1. The maximum knapsack weight capacity was 2, so the maximum possible value was 4. Hence the number of binary variables is equal to 5. To compare the results of all the presented approaches, the same local minimizer was used (the Scipy implementation of L-BFGS-B). The sampling range for angles $\boldsymbol{\gamma}, \boldsymbol{\beta} \in [0, 2\pi]$, and for weights $\alpha_1, \alpha_2 \in [1, 10]$, where $\alpha_1, \alpha_2 \in \mathbb{R}$. We set the number of layers in each variational algorithm to $p = 5$.

Random Search tests were performed by sampling 10.000 initial points (i.e., variational parameter initial sets) and then calculating the results of Algorithm 2 for each point. In particular, as we wanted to check the performance against the number of variational algorithm (QAOA, WS-WAOA or H-QAOA) launches, the results were obtained as follows ($n$ – number of launches of a particular QAOA-based variant):

- shuffle set of 10.000 points,
- split initial set into 10.000/$n$ groups, each containing $n$ points,
- for each group, find the minimum value in this group,
- calculate mean and standard deviation for the 10.000/$n$ minimum values.

Tests for Basin-hopping (see detailed parameters in Tab. 1) were performed with different numbers of starting points, keeping the number of variational algorithm launches the same as for Random Search. For Basin-hopping the choice of the initial point is important as it determines the next steps of the algorithm (see *random_step()* in Algorithm 3). Therefore, in this paper, we also compare the results of splitting the global optimizer run into several independent trials with different initial points. These configurations were: (10, 1), (10, 5), (10, 10), (10, 20), (10, 25), (10, 50), (10, 100), (50, 1), (50, 2), (50, 4), (50, 5), (50, 10), (50, 20), (100, 1), (100, 2), (100, 5), (100, 10), (200, 1), (200, 5), (250, 1), (250, 2), (250, 4), (500, 1), (500, 2), (1000, 1), where the first number indicates the number of iterations per initial point, and the second one indicates the number of initial points. Each configuration was tested 10 times and then the mean and standard deviation were calculated. All calculations were performed on Ares Supercomputer (ACC Cyfronet AGH) with an Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz.

Table 1: Basin-hopping parameters. $N$ is the number of iterations per init point. The description of each parameter can be found in the SciPy documentation.

| **Basin-hopping parameters** | | | | |
|---|---|---|---|---|
| $x_0$ | niter | T | stepsize | interval |
| init point | $N$ | 1 | 0.5 | 50 |

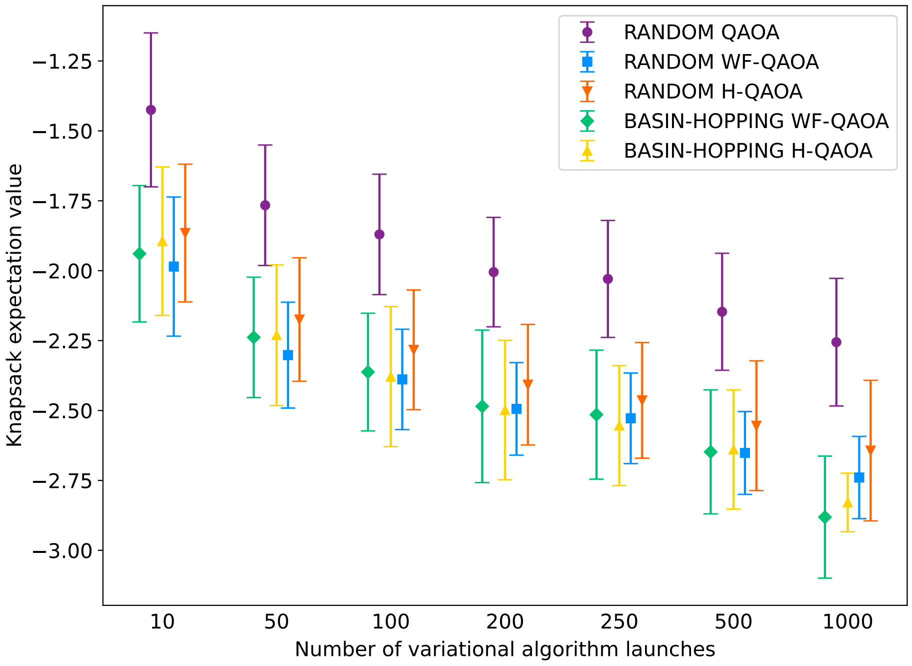


Fig. 3: Negated knapsack expectation value calculated by Algorithm 1 with standard deviation (standard error, normalized by $N-1$). For Basin-hopping only the best configuration for each number of launches was chosen (the exact ratio of initial points and number of iterations can be found in Tab. 2). The lower the expectation value, the better the quality of the results.

Table 2: The detailed results from Fig 3 with Basin-hopping configuration of number of iterations (iters) and number of initial points (init).

| Number of launches | **Basin-hopping** | | | | | | **Random Search** | | |
|---|---|---|---|---|---|---|---|---|---|
| | H-QAOA | | | WF-QAOA | | | | | |
| | iters | inits | result | iters | inits | result | H-QAOA | WF-QAOA | QAOA |
| 10 | 10 | 1 | -1.89±0.26 | 10 | 1 | -1.93±0.24 | -1.87±0.24 | -1.99±0.24 | -1.42±0.27 |
| 50 | 10 | 5 | -2.21±0.25 | 10 | 5 | -2.23±0.21 | -2.17±0.21 | -2.29±0.20 | -1.76±0.22 |
| 100 | 10 | 10 | -2.37±0.25 | 10 | 10 | -2.36±0.21 | -2.29±0.22 | -2.39±0.17 | -1.89±0.20 |
| 200 | 10 | 20 | -2.49±0.25 | 200 | 1 | -2.48±0.27 | -2.39±0.22 | -2.51±0.17 | -1.99±0.21 |
| 250 | 10 | 25 | -2.56±0.22 | 50 | 5 | -2.50±0.19 | -2.43±0.22 | -2.52±0.17 | -2.02±0.20 |
| 500 | 250 | 2 | -2.69±0.16 | 10 | 50 | -2.63±0.23 | -2.55±0.22 | -2.62±0.16 | -2.13±0.22 |
| 1000 | 10 | 100 | -2.80±0.17 | 200 | 5 | -2.85±0.24 | -2.65±0.22 | -2.71±0.17 | -2.24±0.22 |

Fig. 3 and the corresponding Tab. 2 indicate that with increasing number of launches of the QAOA-based variants, the results improve, which is expected. We can observe that all the proposed modifications performed better than the regular QAOA. For lower number of variational algorithm launches, the best results were achieved by Random Search with WF-QAOA. On the contrary, Basin Hopping variants took advantage of higher number of launches, especially with more initial points (see Tab. 2 for details) and performed slightly better. In general, the difference between results of Basin-hopping H-QAOA and WF-QAOA extensions is very small and requires further investigation.

## 8 Summary and Future Work

In this paper, we proposed two variants of QAOA modifications dedicated to constrained combinatorial problems. The first one is based on using weight-free Hamiltonian as the observable. The second method further extends this idea by adding QUBO weights to the set of variational parameters. We used the presented QHyper framework to compare proposed modifications with the original algorithm on the example of the Knapsack Problem. Additionally, we compare the sampling of the initial variational parameters using two global optimizers: a simple Random Search and a more sophisticated algorithm — Basin-hopping. All of the proposed modifications outperform the original QAOA algorithm. Due to the flexibility of the QHyper API, we were able to quickly evaluate various approaches with different configurations. Additionally, the modular architecture of QHyper should allow for extension to repeat experiments for a new problem.

Our experiments demonstrate that using the proposed weight-free Hamiltonian (see Section 4) to calculate the expectation value of the problem cost function can have a significant impact on the outcomes. This is feasible in gate-based variational algorithms, where the Hamiltonian used for the ansatz problem encoding does not have to be identical to the actual observable. What is more, estimating the expectation value can be performed using a classical algorithm based on a fixed number of measurement results (see Algorithm 1). On the contrary, the solution cannot be easily transformed to quantum annealers, where the Ising formulation of the problem Hamiltonian is required.

The next step will be to further investigate the impact of the Basin-hopping method on the results as well as the usefulness of other global optimizers such as the CEM. We also plan to try out the presented approach on larger Knapsack Problem instances and for different problems like the Traveling Salesman Problem or the Workflow Scheduling Problem.

## Acknowledgements

This work was supported (in part) by the EuroHPC PL infrastructure funded at the Smart Growth Operational Programme (2014-2020), Measure 4.2 under the grant agreement no. POIR.04.02.00-00-D014/20-00. We gratefully acknowledge Poland's high-performance computing infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2023/016301. The authors are grateful for support from the subvention of Polish Ministry of Education and Science assigned to AGH University of Science and Technology (Faculty of Computer Science, Electronics and Telecommunications).

**Code available at** github.com/qc-lab/QHyper.

## References

1. Ayodele, M.: Penalty Weights in QUBO Formulations: Permutation Problems. In: Evolutionary Computation in Combinatorial Optimization, vol. 13222, pp. 159–174. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-04148-8_11, https://link.springer.com/10.1007/978-3-031-04148-8_11, series Title: Lecture Notes in Computer Science
2. Bärtschi, A., Eidenbenz, S.: Grover Mixers for QAOA: Shifting Complexity from Mixer Design to State Preparation. In: 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). pp. 72–82 (Oct 2020). https://doi.org/10.1109/QCE49297.2020.00020, http://arxiv.org/abs/2006.00354, arXiv:2006.00354 [quant-ph]
3. Endres, S.C., Sandrock, C., Focke, W.W.: A simplicial homology algorithm for Lipschitz optimisation. Journal of Global Optimization **72**(2), 181–217 (Oct 2018). https://doi.org/10.1007/s10898-018-0645-y, http://link.springer.com/10.1007/s10898-018-0645-y
4. Farhi, E., Goldstone, J., Gutmann, S.: A Quantum Approximate Optimization Algorithm (Nov 2014), http://arxiv.org/abs/1411.4028, arXiv:1411.4028 [quant-ph]

5. Fuchs, F.G., Lye, K.O., Nilsen, H.M., Stasik, A.J., Sartor, G.: Constrained mixers for the quantum approximate optimization algorithm. Algorithms **15**(6), 202 (Jun 2022). `https://doi.org/10.3390/a15060202`, `http://arxiv.org/abs/2203.06095`, arXiv:2203.06095 [quant-ph]
6. Golden, J., Bärtschi, A., Eidenbenz, S., O'Malley, D.: Evidence for Super-Polynomial Advantage of QAOA over Unstructured Search (Feb 2022), `http://arxiv.org/abs/2202.00648`, arXiv:2202.00648 [quant-ph]
7. Hadfield, S., Wang, Z., O'Gorman, B., Rieffel, E.G., Venturelli, D., Biswas, R.: From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. Algorithms **12**(2), 34 (Feb 2019). `https://doi.org/10.3390/a12020034`, `http://arxiv.org/abs/1709.03489`, arXiv:1709.03489 [quant-ph]
8. Hadfield, S., Wang, Z., Rieffel, E.G., O'Gorman, B., Venturelli, D., Biswas, R.: Quantum Approximate Optimization with Hard and Soft Constraints. In: Proceedings of the Second International Workshop on Post Moores Era Supercomputing. pp. 15–21. ACM, Denver CO USA (Nov 2017). `https://doi.org/10.1145/3149526.3149530`, `https://dl.acm.org/doi/10.1145/3149526.3149530`
9. Lucas, A.: Ising formulations of many NP problems. Frontiers in Physics **2** (2014). `https://doi.org/10.3389/fphy.2014.00005`, `http://arxiv.org/abs/1302.5843`, arXiv:1302.5843 [cond-mat, physics:quant-ph]
10. Mesman, K., Al-Ars, Z., Möller, M.: QPack: Quantum Approximate Optimization Algorithms as universal benchmark for quantum computers (Apr 2022), `http://arxiv.org/abs/2103.17193`, arXiv:2103.17193 [quant-ph]
11. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research and Financial Engineering, Springer New York (2006). `https://doi.org/10.1007/978-0-387-40065-5`, `http://link.springer.com/10.1007/978-0-387-40065-5`
12. Olson, B., Hashmi, I., Molloy, K., Shehu, A.: Basin Hopping as a General and Versatile Optimization Framework for the Characterization of Biological Macromolecules. Advances in Artificial Intelligence **2012**, 1–19 (Dec 2012). `https://doi.org/10.1155/2012/674832`, `https://www.hindawi.com/journals/aai/2012/674832/`
13. Parizy, M., Kakuko, N., Togawa, N.: Fast Hyperparameter Tuning for Ising Machines (Nov 2022), `http://arxiv.org/abs/2211.15869`, arXiv:2211.15869 [cs]
14. Radha, S.K.: Quantum constraint learning for quantum approximate optimization algorithm (Dec 2021), `http://arxiv.org/abs/2105.06770`, arXiv:2105.06770 [physics, physics:quant-ph]
15. Roch, C., Impertro, A., Phan, T., Gabor, T., Feld, S., Linnhoff-Popien, C.: Cross Entropy Hyperparameter Optimization for Constrained Problem Hamiltonians Applied to QAOA (Aug 2020), `http://arxiv.org/abs/2003.05292`, arXiv:2003.05292 [quant-ph]
16. Tilly, J., Chen, H., Cao, S., Picozzi, D., Setia, K., Li, Y., Grant, E., Wossnig, L., Rungger, I., Booth, G.H., Tennyson, J.: The Variational Quantum Eigensolver: A review of methods and best practices. Physics Reports **986**, 1–128 (Nov 2022). `https://doi.org/10.1016/j.physrep.2022.08.003`, `https://linkinghub.elsevier.com/retrieve/pii/S0370157322003118`
17. Tomasiewicz, D., Pawlik, M., Malawski, M., Rycerz, K.: Foundations for Workflow Application Scheduling on D-Wave System. In: Computational Science – ICCS 2020, vol. 12142, pp. 516–530. Springer International Publishing, Cham (2020). `https://doi.org/10.1007/978-3-030-50433-5_40`, `http://link.`

springer.com/10.1007/978-3-030-50433-5_40, series Title: Lecture Notes in Computer Science

18. Wales, D.J., Doye, J.P.K.: Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. The Journal of Physical Chemistry A **101**(28), 5111–5116 (Jul 1997). https://doi.org/10.1021/jp970984n, https://pubs.acs.org/doi/10.1021/jp970984n
19. Wang, Z., Rubin, N.C., Dominy, J.M., Rieffel, E.G.: $XY$-mixers: analytical and numerical results for QAOA. Physical Review A **101**(1), 012320 (Jan 2020). https://doi.org/10.1103/PhysRevA.101.012320, http://arxiv.org/abs/1904.09314, arXiv:1904.09314 [quant-ph]