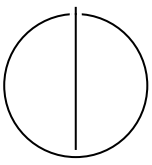
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

Designing Quantum Optimization Algorithms using Random Key Optimization

Alexander Treml



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

Designing Quantum Optimization Algorithms using Random Key Optimization

Entwurf von Quantenoptimierungsalgorithmen mithilfe von Random Key Optimization

Author:	Alexander Treml
Examiner:	Prof. Dr. Christian Mendl
Supervisor:	Jernej Rudi Finzgar
Submission Date:	30.03.2025

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

Munich, 30.03.2025 Alexander Treml

Acknowledgments

I thank Martin Schütz for helpful discussion, and Felix Huber for providing technical support, while he himself was in the middle of writing his thesis.

Abstract

We investigate the efficacy of the random key optimization (RKO) framework presented by Chaves et al. [14] and the applicability of the RKO paradigm in general in areas related to quantum optimization algorithms. We show that the RKO framework is applicable to the task of optimizing the parameters of the Quantum Approximate Optimization Algorithm (QAOA). We also propose a method for using RKO to learn new ansatz structures. We find that the algorithm learns to prefer some ansatz structures over others, but do not observe any performance improvements. We also propose an RKO inspired method for approximately solving the traveling salesman problem (TSP) using a variational quantum algorithm, which only uses linearly many qubits or less but does not produce competitive results.

Contents

1. Introduction

Combinatorial optimization problems (COPs) are ubiquitous in science and industry. Among others, they have applications in finance [55], robot motion planning [58], and molecular biology [40]. Since many COPs are NP-complete, a large variety of practically relevant problems can be reduced to general and well-studied problems. However, this also means that under the assumption $P \neq NP$, no efficient classical solvers exist for such problems. Consequently, there are ongoing efforts to find new ways to tackle specific types of COPs.

The random key optimization (RKO) paradigm is a paradigm for the design and application of metaheuristics for solving COPs [58]. It is inspired by the Random Key Genetic Algorithm (RKGA) introduced by Bean in 1994 [5]. Its basic premise is to run the optimization process over positions of an n-dimensional unit hypercube, using a decoder to map these positions to solutions to a given optimization problem. Restricting the search space to this hypercube simplifies the implementation of general metaheuristics, as well as the introduction of problem-specific strategies in the decoder. The common solution space also allows different metaheuristics to cooperate during the optimization process [14]. In this thesis, we apply the RKO paradigm, and in particular the RKO framework developed by Chaves et al. [14], to problems related to variational quantum algorithms (VQAs).

VQAs are a class of flexible hybrid classical-quantum algorithms. VQAs are resilient against hardware noise [43, 44], which means they can perform well on noisy intermediate-scale quantum (NISQ) devices [53]. This makes them a promising class of algorithms for near-term practical applications of quantum computing. VQAs utilize a classical optimization routine to optimize the parameters of a quantum circuit in order to create a quantum state with certain properties, depending on the task at hand [11]. For example, a VQA can be used to find the ground state of a given Hamiltonian [53], which is an important problem in quantum chemistry. This technique can also be applied to combinatorial optimization. By encoding the cost function of the COP in a Hamiltonian, the problem of finding the ground state of said Hamiltonian becomes equivalent to solving the COP. This is the approach taken by Farhi et al. [19] when they introduced the Quantum Approximate Optimization Algorithm (QAOA). Since

its introduction, much research has been published on QAOA, covering its theoretical properties, performance in practical scenarios, and issues that limit its current realizations. In this thesis, we want to add to this body of research by investigating the application of the RKO paradigm in the context of QAOA and related areas.

The remainder of the thesis is structured as follows: In Section 2, we provide the necessary background on the topics of combinatorial optimization, RKO, and VQAs. In Section 3, we apply the RKO framework to the task of QAOA parameter optimization for solving the MaxCut problem. In Section 4, we propose a modification of the QAOA that optimizes the ansatz structure in addition to the variational parameters. In Section 5, we then apply the RKO paradigm to solve the traveling salesman problem (TSP) using a VQA before concluding with Section 6.

2. Background

2.1. Combinatorial Optimization and Metaheuristics

A COP (S, C) is a mathematical problem characterized by a cost function $C\colon S \to \mathbb{R}$ over a finite set S, also referred to as the solution space. The goal is to find the optimal solution $s_{min} \in S$

$$s_{min} = \arg\min_{s \in S} C(s) \tag{2.1}$$

Note here that we always formulate optimization problems in terms of minimization. A maximization problem can be converted to a minimization problem by inverting the sign of C. This problem class contains many NP-complete problems, such as the TSP (Section 2.3) or MaxCut (Section 2.2) [25], which are of practical and academic interest [58, 15]. Since all problems in NP can be reduced to any NP-complete problem in polynomial time, a solver for any NP-complete problem represents a universal solver for all problems in NP. However, no polynomial runtime solver is known for any NP-complete problem.

Despite this, state-of-the-art optimization suites like Gurobi [28], or CPLEX [32] employ heuristics to allow solving industry scale problems within acceptable runtimes [14]. Alternatively, one can use problem-specific approximation algorithms, which offer faster results but do not guarantee that the optimal solution is found. The strategies most relevant to this thesis are so-called metaheuristics. The definition of the term metaheuristic has shifted over time, but it generally describes a problem-independent search strategy, which balances global and local search to efficiently explore complex solution spaces [46]. Examples of metaheuristics are genetic algorithms [31], where good solutions "evolve" using mechanisms akin to biological evolution, or Particle Swarm Optimization (PSO) [33], which models the self-organizing behavior of swarms to navigate the solution space.

2.2. MaxCut and Ising Models

The MaxCut problem is a common COP in research on quantum optimization [68, 4], and is also used for the experiments in Section 3 and Section 4. Given a weighted

undirected graph $G = (V, E)$ with n vertices with edge weights w_{ij}, the solution space S_{MC} is the set of all partitions of the vertices V into two subsets. The goal is to maximize the sum of edges between the two subsets. These partitions can be represented by assigning a number $s_i \in \{-1, 1\}$ to each vertex i. The cost function can then be formulated as a function over assignments $s = (s_1, \ldots, s_n)$ [2].

$$C_{MC}(s) = -\sum_{(i,j)\in E} \frac{w_{ij}}{2}(1 - s_i s_j) \tag{2.2}$$

Note the sign of the sum, which is used to translate MaxCut into a minimization problem. If $s_i = s_j$, the corresponding term does not contribute to the sum. If $s_i \neq s_j$, the term contributes exactly w_{ij}. This way of expressing the cost function constitutes a classical Ising model.

Ising models are a key concept in solving COPs on quantum devices. A classical Ising model is a quadratic function defined over vectors of n spins $s \in \{-1, 1\}^n$, with couplings J_{ij} and local fields h_i [42]

$$H(s) = -\sum_{i<j} J_{ij} s_i s_j - \sum_{i=1}^{n} h_i s_i \tag{2.3}$$

With $h_i = 0$ and $J_{ij} = w_{ij}/2$, we see that Equation 2.2 is indeed an example of an Ising model. The optimization problem associated with an Ising model asks us to minimize H. This problem is NP-hard [3]. In fact, since MaxCut is NP-complete, the polynomial reduction presented by Equation 2.2 already proves this. This means that Ising models can be used to represent any problem in NP, with only polynomial overhead in the number of variables.

By replacing every spin s_i of the classical Ising model with the Pauli-Z operator acting on the i-th qubit σ_i^z, we can construct a quantum Hamiltonian H_C, which encodes the cost function of the original problem in its eigendecomposition. The eigenvalues are the possible values of the cost function, and the associated eigenvectors are the corresponding solutions. Due to the physical interpretations of the eigenvalues, they are also referred to as the energy levels of H_C. The solution is then the lowest energy eigenstate, the ground state of H_C. For our MaxCut example, H_C would be defined as follows [68]

$$H_C = -\sum_{(i,j)\in E} \frac{w_{ij}}{2}(\mathbb{1} - \sigma_i^z \sigma_j^z) \tag{2.4}$$

Consequently, Ising models allow us to translate classical problems to quantum systems. Solving the original COP becomes equivalent to finding the ground state s_{min} of the

Hamiltonian H_C. Because of this, we will also refer to H_C as the problem Hamiltonian.

Ising models are closely related to quadratic unconstrained binary optimization (QUBO) [42], sometimes also called unconstrained binary quadratic programming (UQBP) [36]. QUBO problems are defined over binary strings $x \in \{0,1\}^n$ instead of spins. However, by substituting every binary variable x_i with a spin $s_i = 2x_i - 1$, the two definitions can be shown to be equivalent [42, 22].

2.3. The Traveling Salesman Problem

The traveling salesman problem (TSP) is another important NP-complete COP, which we will use in Section 5 to showcase an alternative method for solving COPs on quantum hardware, that does not involve Ising models. Informally, the goal of the TSP is to find the shortest route between a set of cities that visits every city exactly once and returns to the starting point. Formally, given a weighted complete graph $G = (V, E)$ with n vertices and positive edge weights w_{ij}, the goal is to find the Hamiltonian cycle with the lowest sum of edge weights. Therefore, the set of feasible solutions is the set of all permutations of V, and the cost function is the sum of all edge weights along the cycle described by the permutation. As noted above, every NP-complete problem can be formulated as an Ising model. Lucas [42] proposes the following reduction, starting with the QUBO formulation of the Hamiltonian cycle problem H_A

$$H_A(x) = A \sum_{v=1}^{n} (1 - \sum_{j=1}^{n} x_{v,j})^2 + A \sum_{j=1}^{n} (1 - \sum_{v=1}^{n} x_{v,j})^2 \tag{2.5}$$

where the vertices V are numbered $1, \ldots, N$, and $A > 0$ is a constant. If $x_{v,j} = 1$, this indicates that vertex v is visited at position j in the route. The first term of H_A is zero if and only if every vertex is visited exactly once, and positive otherwise. The second term is zero if and only if exactly one vertex is visited for each position j. Consequently, H_A is zero if the assignment x represents a valid Hamiltonian cycle. Otherwise, it represents a penalty term scaled by A. By adding a term H_B, which represents the length of the route, we obtain a QUBO formulation of the TSP [42].

$$H_B = B \sum_{(u,v) \in E} w_{uv} \sum_{j=1}^{n} x_{u,j} x_{v,j+1} \tag{2.6}$$

$$H_C = H_A + H_B \tag{2.7}$$

Here, the constant B must be chosen so that $0 < B \cdot max(w_{uv}) < A$, which ensures that the optimal solution does not violate any constraints [42]. The QUBO problem

can then be transformed into an Ising model with the substitution given above. This formulation uses n^2 spins, which can be reduced to $(n-1)^2$ by fixing the starting point [42]. However, even for moderately sized instances, this surpasses the limits of NISQ-devices. For example, the IBM Heron processor with 133 qubits [45] could only handle instances with up to twelve vertices. In Section 5, we propose a method for approximately solving the TSP using n qubits or less, inspired by the RKO paradigm.

2.4. Random Key Optimization

The RKO paradigm is a paradigm for the design of metaheuristics. It prescribes that the metaheuristic operates on an n-dimensional unit hypercube as a search space. The user needs to supply a problem-specific decoder that maps the unit hypercube to the solution space of the problem. Formally, for a COP (S, C), a decoder is a function $f_D \colon \mathcal{H} \to S$ with

$$\mathcal{H} = \{(\chi_1, \ldots, \chi_n) \in \mathbb{R}^n | \forall i \in [n] \colon \chi_i \in [0, 1)\} \tag{2.8}$$

We refer to $\chi \in \mathcal{H}$ as a key vector and to an element χ_i as a random key, or simply key. Figure 2.1 illustrates this idea. The paradigm was initially introduced by Bean [5] as a mechanism to ensure solution feasibility in the context of genetic algorithms. Especially when the solution space of a COP involves permutations of elements, e.g. in the case of the TSP, it is difficult to define crossover operators that preserve solution feasibility. In contrast, it is trivial to define operators that recombine random key vectors into other valid random key vectors. Hence, Bean proposes to perform all evolutionary operations on the random key vectors and employ a decoder to recover a valid permutation. In the case of the TSP, the decoder returns the permutation induced by sorting the random keys inside a given vector. Since every vector can be sorted, and sorting provides a valid permutation, all elements of $\mathcal{H}$ correspond to feasible solutions. This use of the vector elements as sorting keys is the origin of the terminology "random keys".

The RKO framework developed by Chaves et al.[14] is a hybrid metaheuristic. It leverages the common search space of the RKO paradigm to allow multiple metaheuristics to cooperate on a given problem. We will provide an overview of its core functionality and the features that are relevant to our use case. A more in-depth description is provided in the original report [14], which also links to the code repository. Note that we describe the framework as it was modified for the purposes of this thesis. These modifications are discussed in Section 3.1. The framework conducts a search by running multiple metaheuristics in parallel. The best solutions at each point in time are aggregated in a global solution pool that is shared between metaheuristics. The metaheuristics are

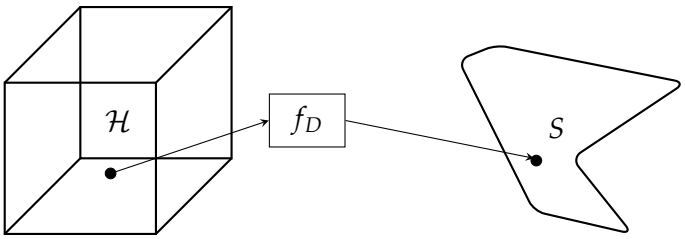


Figure 2.1.: Illustration of the RKO paradigm, adapted from Schütz et al. [58]. The decoder f_D maps a simple point inside the hypercube to a possibly complex solution space.

supplemented by local search algorithms that are used to refine promising solutions in the solution pool or are invoked by the metaheuristics internally. The framework implements the following metaheuristics:

BRKGA The Biased Random Key Genetic Algorithm (BRKGA) [26] is a variant of the RKGA. In each generation, the best p_e individuals are preserved as the "elite". The non-elite individuals are discarded. The now open positions in the population are filled in part with p_m mutants, which are fully random key vectors. The remaining positions are filled by performing parameterized uniform crossover [17] between parents randomly chosen from the previous generation with the inheritance bias ρ. The BRKGA only deviates from the RKGA in the selection of parents for crossover. Here, the first parent is always an elite individual, while the second parent is a non-elite individual.

BRKGA-CS A variant of BRKGA that uses clustering search (CS) to intensify the exploration of promising regions [51].

Simulated Annealing During Simulated Annealing (SA) [35], an initial solution is repeatedly perturbed. If the perturbed solution is better than the previous solution, it is accepted and serves as a new starting point. If it is worse, it may still be accepted depending on the current "temperature". The algorithm starts at a high temperature T_0, where worse solutions are often accepted. After every SA_{max} iterations, the temperature decreases over time with a cooling rate α, reducing the likelihood of accepting worse solutions. In the RKO framework, the perturbations consist of a series of swapping, mirroring, and randomization operations on the random keys, referred to as "shaking". The intensity of the shaking is controlled by the parameters β_{min} and β_{max}.

GRASP The framework implements a Greedy Randomized Adaptive Search Procedure (GRASP) [21] adapted for RKO. First, a list of candidates is generated using line

searches centered on an initial solution. The grid density for these line searches is determined by an internal value h. Then a random candidate is selected to contribute a key for updating the initial solution, which is then fixed. A value α controls the cost range in which candidates are considered, similar to SA. This is repeated until all keys are fixed. The algorithm then tries to improve the solution using local search strategies. The procedure is repeated with decreasing values of h, starting at h_s, until it reaches h_e.

ILS Iterated Local Search (ILS) [41] iteratively perturbs a solution using the shaking method, and then applies local search. The idea is to converge to a local minimum and subsequently escape it in order to find a better solution. The intensity of the shaking is again controlled by the parameters β_{min} and β_{max}.

VNS Variable Neighborhood Search (VNS) [47] also alternates perturbation and local search. In contrast to ILS, it increases the strength of the perturbation with each iteration without improvement. In other words, it increases the size of the neighborhood that is considered during the perturbation step. The intensity of the shaking, meaning the size of the neighborhoods is determined by β_{min}.

PSO PSO [33] is a metaheuristic inspired by swarms of animals. A population of p particles is placed in the search space. In each step, all particles update their velocity based on their individual best known solution, and the globally best known solution of all particles. They then move according to their new velocity. The velocity update is controlled by the particle weight w, and the coefficients c_1 and c_2 which control the contribution of local and global information respectively.

SGA A standard genetic algorithm [31] adapted to RKO using tournament selection with population size p. During crossover, parents exchange individual keys with a probability p_c. Random mutations are also applied to the keys during crossover with probability μ.

LNS Large Neighborhood Search (LNS) [57] works by destroying an existing solution and subsequently repairing it. In the case of the RKO framework, this means removing a number of keys and replacing them with keys found through greedy iterated line searches. The algorithm takes the parameters β_{min} and β_{max}, which control how many keys are removed, and T_0 and α which determine the chance of a solution being accepted, similar to SA

The local search procedure implemented in the RKO framework is Random Variable Neighborhood Descent (RVND) [52]. The procedure explores multiple predefined neighborhood structures in random order. The available structures are "swap", where keys are swapped inside a vector, "invert", where a key χ_i is replaced by $1 - \chi_i$, and Nelder-Mead, which generates candidates based on a modified version of the Nelder-Mead minimization algorithm [49]. It is noteworthy that none of these neighborhoods is defined using the Euclidean or any other geometric distance of key vectors. In fact, apart from PSO, no metaheuristic applies continuous changes to the key vectors. Notions of distance in the framework rather resemble edit distances, with "swap", "replace" and "mirror" as possible edits. This highlights that the framework was developed with combinatorial optimization in mind, which is discrete in nature.

A full run of the RKO framework runs all nine metaheuristics in parallel. In our experiments, we will refer to this setup simply as "RKO". Alternatively, one can run nine instances of the same method in parallel. This can be used to determine, whether the diversity of heuristics contributes to the performance of the framework, or if a single method dominates for a given problem. We will refer to these setups as "RKO-X", where "X" is the respective metaheuristic.

2.5. Variational Quantum Algorithms

We will now give a general overview of VQAs. A more extensive review of VQAs and related research is provided by Cerezo et al. [11]. A VQA mainly consists of three components, namely a circuit ansatz, a classical optimization routine, and a cost function. The goal of the algorithm is to prepare a quantum state that is related to the solution of a given problem. For example, in the context of combinatorial optimization, a standard basis measurement on the output state would ideally yield a binary string that solves the COP.

The circuit ansatz describes the quantum operations that are used to prepare the output state. More precisely, it provides a unitary $U(\theta)$, parameterized by the vector θ of angle parameters $\theta_1, \cdots, \theta_n$. We define the output state $|\psi(\theta)\rangle$ for some input state $|\psi_0\rangle$:

$$|\psi(\theta)\rangle = U(\theta)\,|\psi_0\rangle \tag{2.9}$$

The cost function $f_C(\theta)$ provides a metric for the quality of $|\psi(\theta)\rangle$. For example, a natural choice for combinatorial optimization is the expectation value of the problem

Hamiltonian H_C introduced in Section 2.1. The value of $f_C(\theta)$ then corresponds to the expectation value of the cost function $\mathbb{E}(C(s))$, if s is the result of a standard basis measurement on $|\psi(\theta)\rangle$. It follows that measurements will tend to yield better solutions s if $f_C(\theta)$ is low.

$$f_C(\theta) = \langle\psi(\theta)| H_C |\psi(\theta)\rangle = \mathbb{E}(C(s)) \tag{2.10}$$

Other choices for f_C are possible. Barkoutsos et al. [4] argue that the expectation value does not accurately reflect the quality of a solution in a practical setting. When applying a VQA to optimization, only the best result out of multiple measurements is relevant. Suppose there are two sets of parameters θ_A and θ_B that produce equal expectation values

$$\langle\psi(\theta_A)| H_C |\psi(\theta_A)\rangle = \langle\psi(\theta_B)| H_C |\psi(\theta_B)\rangle = 0.5 \tag{2.11}$$

Measurements on $|\psi(\theta_A)\rangle$ always yield a solution with cost 0.5, while measurements on $|\psi(\theta_B)\rangle$ yield solutions with cost 0 or 1 with equal probability. Clearly, θ_B is preferred in practice, since with only a few repeated measurements, it will yield a better solution than θ_A. To account for this, they recommend the conditional value-at-risk (CVaR) cost function defined as

$$CVaR_\alpha(X) = \frac{1}{\lceil \alpha k \rceil} \sum_{i=0}^{\lceil \alpha k \rceil} X_i \tag{2.12}$$

where $X = (X_0, \ldots, X_k)$ is a vector of samples sorted in ascending order. Essentially, $CVaR_\alpha(X)$ is the average over the best $\lceil \alpha k \rceil$ samples. This would cause the optimizer to prefer θ_B over θ_A. Because of this advantage, we will use the CVaR cost function for the experiments in Sections 3 and 4.

Finally, the goal of the classical optimization routine is to select the optimal parameters θ_{min}

$$\theta_{min} = \arg\min_\theta f_C(\theta) \tag{2.13}$$

A good choice of classical optimizer is an active area of research. For many common cost functions, the gradient can be accessed using so-called parameter-shift rules [59], or estimated using finite difference methods. This enables the use of gradient-based optimizers such as Adam [34, 27] or Simultaneous Perturbation Stochastic Approximation (SPSA) [61, 62]. However, the cost landscapes of VQAs typically contain many local minima, which can trap gradient-based methods. Barren plateaus are also a common feature of these landscapes [39, 65], where the gradient vanishes exponentially with the size of the system. This reduces the trainability of the ansatz when using gradient-based methods [39]. For these reasons, derivative-free methods like Constrained Optimization By Linear Approximation (COBYLA) [54, 4] or metaheuristics like PSO [37] have also

seen application in VQAs. In Section 3 we evaluate the efficacy of the RKO framework for this task.

2.6. The Quantum Approximate Optimization Algorithm

The QAOA is a concrete instance of a VQA designed for approximately solving COPs [19]. For the QAOA, the initial state is set to equal-superposition state $|+\rangle^{\otimes n}$ where n is the number of qubits. The circuit ansatz of the QAOA then alternately applies the problem Hamiltonian H_C and a mixer Hamiltonian H_B across p layers. The corresponding operators are the unitary time evolution operators $U_H(t) = e^{-itH}$ for a time t and a Hamiltonian H. For QAOA, these operators are parameterized with the angle parameters γ and β respectively. The final state $|\psi(\theta)\rangle$ is then given by

$$|\psi(\theta)\rangle = U_{H_B}(\beta_p)U_{H_C}(\gamma_p)\cdots U_{H_B}(\beta_1)U_{H_C}(\gamma_1)\,|+\rangle^{\otimes n} \tag{2.14}$$

where $\theta = (\gamma_1, \beta_1, \cdots, \gamma_p, \beta_p)$ and the mixer Hamiltonian defined as

$$H_B = \sum_{j=1}^{n} \sigma_j^x \tag{2.15}$$

where σ_j^x is the Pauli-X gate acting on the j-th qubit. The cost function is simply the expectation value of H_C, as described above. For fixed p, Farhi et al. provide an analytical method for determining the optimal θ_{min}, which has polynomial runtime for problems defined over graphs of bounded degree [19]. This method leans on the fact that for fixed p, the state of each qubit in the output state can only depend on a certain subset of qubits [19]. Taking the MaxCut example, in each layer, information can only be exchanged between qubits that share an edge in the underlying graph, via their connection through a quadratic term in H_C. Consequently, after p layers, each qubit can only "see" qubits that are within p edges. This property is referred to as "locality" and while it enables efficient parameter selection for low p, it can be shown to limit the performance of the QAOA [9, 18]. In practice, one typically uses numerical optimization techniques as for other VQAs, since the complexity of this method grows exponentially with p [68].

Finally, it is worth noting that the QAOA is closely connected to the Quantum Adiabatic Algorithm (QAA), introduced by Farhi et al. [20]. The adiabatic theorem [8] states that a quantum system in an instantaneous eigenstate of a time-dependent Hamiltonian $H(t)$ will stay in that eigenstate if the Hamiltonian changes sufficiently slowly, and the energy gap between the current state and any other state never closes.

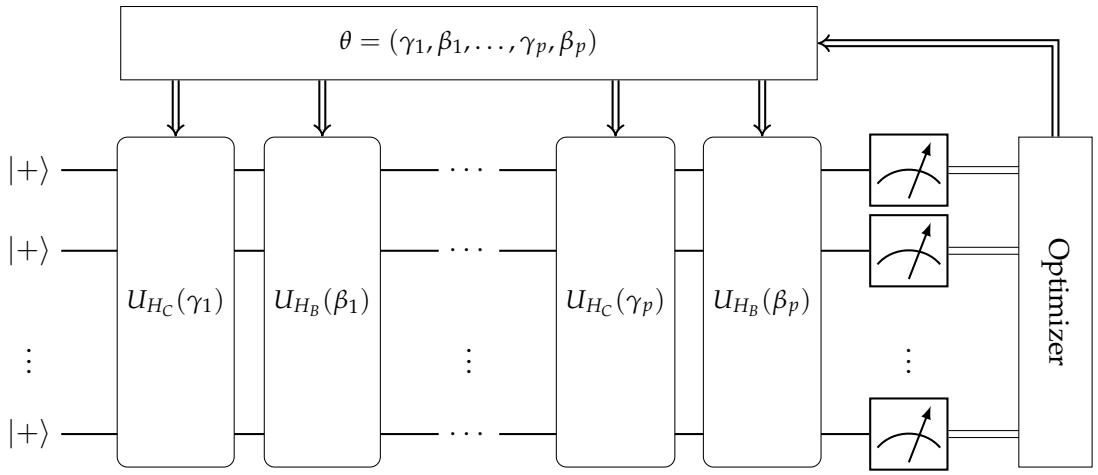


Figure 2.2.: QAOA schematic adapted from Zhou et al.[68]

In particular, if the system starts in the ground state of a Hamiltonian H_B, which is slowly transformed into a different Hamiltonian H_C while maintaining a sufficient energy gap to the first excited state, then the system will end in the ground state of H_C. This is the principle employed by the QAA to solve COPs. H_B is chosen such that its ground state is easily determined, e.g., as for the QAOA with the ground state $|+\rangle^{\otimes n}$. H_C is used to encode the solution as above, e.g., using an Ising model. The QAA can then obtain the solution by applying the time-dependent Hamiltonian $H(t)$

$$H(t) = (1 - \frac{t}{T})H_B + \frac{t}{T}H_C \tag{2.16}$$

where T is the overall evolution time. Note that at $H(0) = H_B$ and $H(T) = H_C$. To ensure that the adiabatic theorem is applicable, T must be large enough. Otherwise, the system might transition into a higher energy state. In general, $T \propto \frac{1}{\Delta^2}$, where Δ is the minimum energy gap [20], i.e., the energy difference between the first excited state and the ground state. For a fixed point in time t' and $\tau = \frac{t'}{T}$, the unitary $U_H(t') = e^{-1H(t')}$ can be expressed using the Suzuki-Trotter decomposition [63] as

$$U_H(t') = e^{-1H(t')} = \lim_{n\to\infty}(e^{-i(1-\tau)/nH_B}e^{-i\tau/nH_C}) \tag{2.17}$$

For $n = 1$, this corresponds to one layer of the QAOA ansatz with the angle parameters $\beta = (1 - \tau)$ and $\gamma = \tau$. Therefore, the QAOA can be seen as a trotterized approximation of the QAA [19].

3. RKO for QAOA-Training

In this section, we assess the performance of the RKO framework as a classical optimizer for training QAOA parameters. Additionally, these results will serve as a baseline to see if the modifications in Section 4 can improve performance. The task is to optimize the QAOA parameters for 100 problem instances for $p = 3$ and $p = 6$. Before presenting our results, we outline the general methodology that serves as the basis for this experiment, and the followup experiment in Section 4.

3.1. Choice of Optimizers and Configuration

The focus of these experiments is the performance of the RKO framework as a whole. Additionally, we compare the performance of the individual metaheuristics employed by the RKO framework, similar to the analysis by Chaves et al. [14]. To establish a baseline, we use COBYLA and SPSA, which are other popular optimizers for VQA tasks [4, 12, 23, 62]. We use the COBYLA implementation provided by the SciPy library [64] and the SPSA implementation provided by the Pennylane library. Finally, we also compare the results to pure random sampling, which we refer to as "RAND". This should establish whether the optimizers are principally able to navigate the cost landscape.

We adjusted the RKO framework to be better suited for our experiments. Firstly, since our analysis focuses on the number of cost function evaluations instead of wall clock time as a performance measure, we replaced the time-based stopping criterion of the framework with a budget of decoder calls. To minimize the impact of thread scheduling, we distribute the budget equally over all metaheuristics running in parallel. The budget is set to 9000 decoder calls for all experiments, 1000 calls for each thread. Because this number of decoder calls is relatively low, we also disabled the "FareyLS" local search procedure of the framework. This method was used to conduct an extensive search to improve the pool of initial solutions but would use up a large part of the evaluation budget in the process.

Since the RKO framework, as a metaheuristic, is fundamentally different from COBYLA and SPSA, setting up a meaningful comparison is non-trivial. By running

multiple routines in parallel, the RKO framework has an inherent mechanism for diversifying starting conditions. In contrast, the performance of a single COBYLA or SPSA run could be impacted by an unfavorable starting point. A solution would be to also distribute the budget of 9000 calls over nine restarts of the optimizer. However, in our initial experiments, both COBYLA and SPSA would converge well before using 1000 calls. This would then cause these methods to underutilize their resources and skew the results towards the RKO framework. Instead, we opted to implement restarting mechanisms that allow both optimizers to use the full budget effectively. For COBYLA, a restart is triggered when the trust region shrinks below the size specified by the hyperparameter *tol*. SPSA restarts when no improvement has been achieved for a number of iterations, controlled by the *patience* hyperparameter.

3.2. Cost Function and Performance Metrics

We use the CVaR cost function for our experiments, as described in Section 2.5. We set $\alpha = 0.25$ as recommended by Barkoutsos et al. [4]. We compute the cost function from the samples produced by $k = 10000$ shots of the circuit. This also introduces a stochastic element, which is inherent to quantum measurements. Accounting for this is important for the comparison of optimizers for VQAs in a realistic setting [62].

To assess the final quality of the optimization results, CVaR is unsuitable because it is only indirectly related to quantities of interest. Since all our experiments are conducted using simulated devices provided by the Pennylane Python library [6], we have direct access to the probability of measuring the optimum p_{opt}, which we can use as a metric of solution quality. This is also in line with the approach by Barkoutsos et al. [4], offering some comparability. We also compare the number of solved instances for each method. In a practical setting, a problem instance is solved if the optimal solution is measured at any point. However, since our test instances are small and the shot counts large, even uniform random sampling could reliably produce the optimal solution at least once. Therefore, for the purposes of this experiment, we consider an instance "solved" if p_{opt} exceeds 0.1.

3.3. Hyperparameter Optimization

All optimizers in our experiments have hyperparameters that can heavily impact their performance. For a fair comparison, it is therefore crucial to select the right sets of hyperparameters [62]. We choose the best set of parameters out of 40 trials. A trial consists of multiple optimization runs on different problem instances, with a budget

of 1000 decoder calls. The average probability of measuring the optimum over all instances is returned. Instances used for hyperparameter optimization are not used for subsequent evaluation. The sets of parameters are sampled using the TPE sampler provided by the Optuna framework [1] with default settings. For the RKO framework, hyperparameter optimization is performed separately for each metaheuristic without running multiple instances in parallel. The hyperparameters chosen for each individual metaheuristic are then combined to obtain the full parameterization of the framework. Hyperparameter optimization was performed on five instances of size $n = 8$ for each value of p separately. Table A.1 in Appendix A contains the chosen parameters.

3.4. Statistical Tests

For comparing the set of methods, we use a Friedman test [24] followed by a Nemenyi test [50] for post-hoc analysis, analogous to the analysis in the original publication on the RKO framework [14]. These tests are available in the SciPy library [64]. The Friedman test is a non-parametric test for comparing multiple sets of dependent samples. As such, it does not depend on normally distributed samples. It rejects the null hypothesis H_0 if there are statistically significant performance differences within the set of tested methods. In the context of comparing k algorithmic methods on n problem instances, the test involves the following steps [24]:

1. Arrange the test results in a table, with the columns corresponding to the tested methods and rows corresponding to the problem instances.
2. In each row, rank the results from 1 to k.
3. Compute the mean rank $\bar{r}_j$ of all columns j. Under the assumption of H_0, the mean rank should be similar between the columns. A significant deviation would indicate that one method performs better/worse than the others.
4. Compute χ_r^2 according to Equation 3.1. If H_0 holds, and k and n are sufficiently large, this value would be sampled from the χ_k^2 distribution. Thus the corresponding percentile provides a measure of significance.

$$\chi_r^2 = \frac{12n}{p(p+1)} \sum_{j=1}^{p} (\bar{r}_j - \frac{1}{2}(p+1))^2 \tag{3.1}$$

We then use the Nemenyi test for pairwise analysis of the methods, which tests if the difference between the mean ranks of two methods falls outside a confidence interval, depending on the significance level α [50]. We set $\alpha = 0.05$ for both tests.

3.5. Problem Instances

As problem instances, we generate random weighted three-regular graphs (w3Rs) with edge-weights chosen uniformly from the interval $[0, 1)$. Only connected instances are considered. This type of graph is commonly used for benchmarking QAOA [4, 68], and thus provides some comparability. We use graphs with 8, 10, 12, and 14 nodes, for which the optimal solutions were calculated using the SCIP optimization suite [7]. The test set consists of 25 instances of each size, 100 instances in total.

3.6. Parameter Decoder

Finally, the decoder for this problem simply multiplies the key vector by 2π to retrieve the angle parameters θ. This restricts each θ_i to the range $[0, 1)$. However, according to Zhou et al. [68], the optimal QAOA parameters for w3Rs of our specification tend to lie roughly between 0.05π and 0.75π. Therefore, the restricted parameter space should not limit the optimizers. For $p = 3$ and $p = 6$, the ansatz has six and twelve free parameters, respectively, requiring key vectors of length $n_k = 6$ and $n_k = 12$.

3.7. Results

We begin with the results for $p = 3$. The Friedman test rejects the null hypothesis with $p = 1.47e - 188$. Figure 3.1 shows the result of the Nemenyi test. The significance plot shows that most pairs of methods are distinguishable from each other. Some RKO methods show no significant differences between each other, but no clear pattern is visible. Surprisingly RKO-SA and SPSA can not be distinguished from RAND. To qualitatively assess the differences, the probability of measuring the optimum solution p_{opt} aggregated over instances of size $n = 14$ for each algorithm is plotted in Figure 3.2. Here, two groups emerge. RAND, RKO-SA, and SPSA perform similarly, and clearly worse than all other methods. The remaining methods perform roughly equally well, with COBYLA performing worse on average, but reaching the same maximum value as most RKO methods.

On the left-hand side of Figure 3.3, the difficulty of the instances in the test set is visualized, by plotting the average p_{opt} achieved by all algorithms, with error bars marking the minimum and maximum values observed. The instances are separated by problem size, and sorted by decreasing p_{opt}, indicating increasing difficulty. This shows that the difficulty distribution of the test set is not strictly clustered around the problem sizes but instead is spread evenly. Significant overlap exists between different

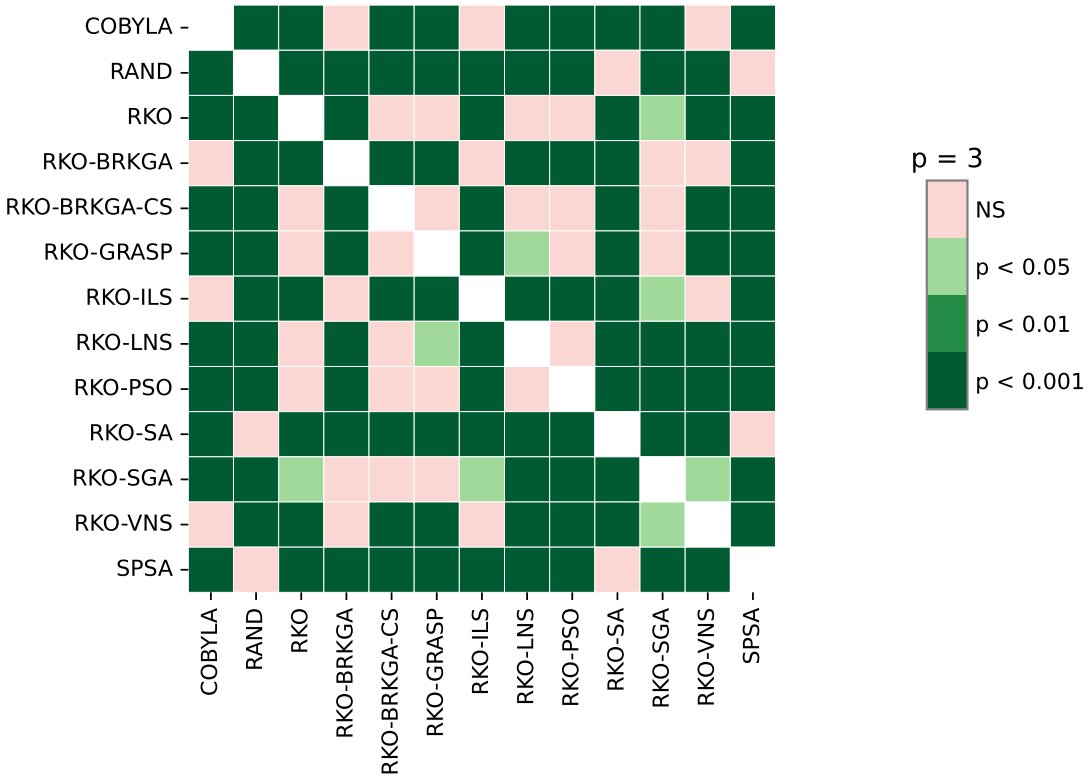


Figure 3.1.: The results of the Nemenyi test for QAOA parameter optimization ($p = 3$). Most pairs of methods can be distinguished. RAND, RKO-SA, and SPSA show no significant differences between each other.

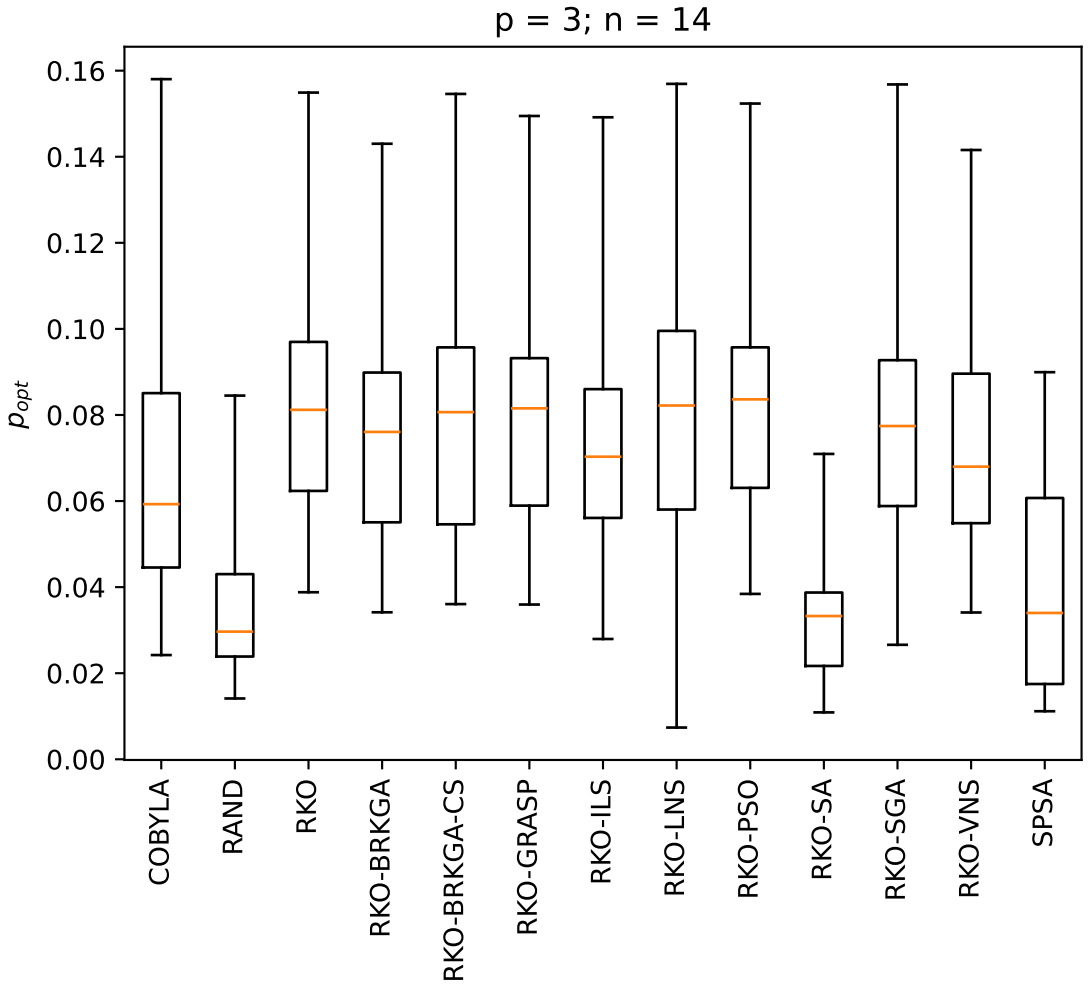


Figure 3.2.: Probability of measuring the optimum solution p_{opt} for each method, aggregated over problem instances of size $n = 14$ ($p = 3$). The boxes indicate the interquartile range. The whiskers cover the whole range of observed values. The orange line indicates the median.

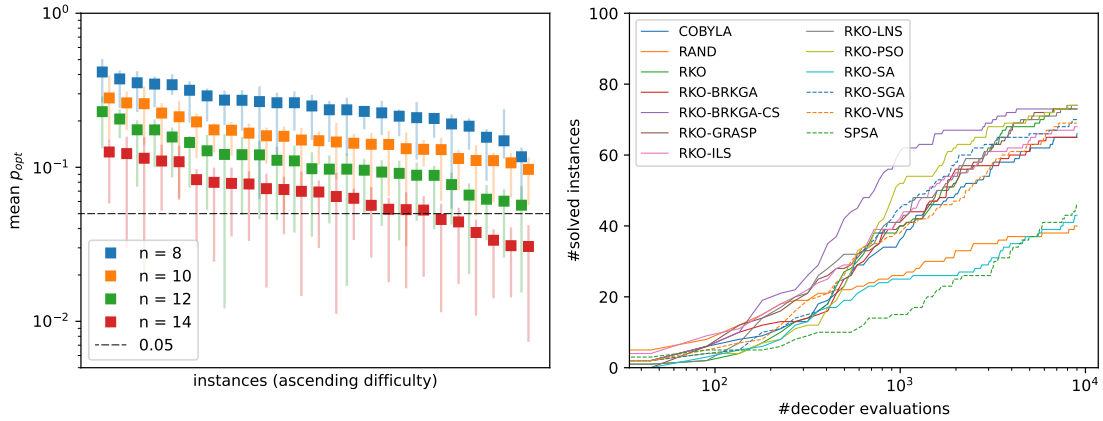


Figure 3.3.: On the left, mean value of p_{opt} per test instance. Instances are sorted by ascending difficulty (decreasing p_{opt}) within each problem size. Error bars indicate maximum and minimum values. Horizontal positions are offset for readability and have no further meaning. On the right, the number of solved instances ($p_{opt} > 0.1$) for $p = 3$ plotted against the number of decoder evaluations on a logarithmic scale.

problem sizes, e.g. the hardest instance of size $n = 8$ is harder than the easiest instance of size $n = 12$. We make use of this fact to investigate the convergence behavior of the algorithms, by plotting the number of solved instances against the number of decoder evaluations (See Figure 3.3 on the right). Since the difficulties are distributed evenly, the curves do not contain any sudden jumps. Just as the algorithms exhibit no major differences in final outcomes, all algorithms except for RAND, RKO-SA, and SPSA converge at similar rates. Only RKO-BRKGA-CS is continuously ahead up until 3000 function evaluations. Overall, the maximum number of instances solved by any method is 74. RKO, RKO-PSO, and RKO-BRKGA-CS tie under this metric.

We now move to the results for $p = 6$. The Friedman test again rejects the null hypothesis with $p = 1.85e - 124$. The significant differences found by the Nemenyi test in Figure 3.4 are more sparse in this setting. Many methods are no longer distinguishable. This is likely due to the fact, that most methods achieve values $p_{opt} > 0.25$ for the easier instances, at which point the CVaR with $\alpha = 0.25$ does not reward any further improvement. However, SPSA is now distinguishable from RAND. Figure 3.5 again depicts the distribution of p_{opt} for each method on instances of size $n = 14$. The orange dots now mark the previous median for $p = 3$. COBYLA, RAND, RKO-SA, and SPSA perform worse than for $p = 3$, while all other methods have improved. RKO-SGA and

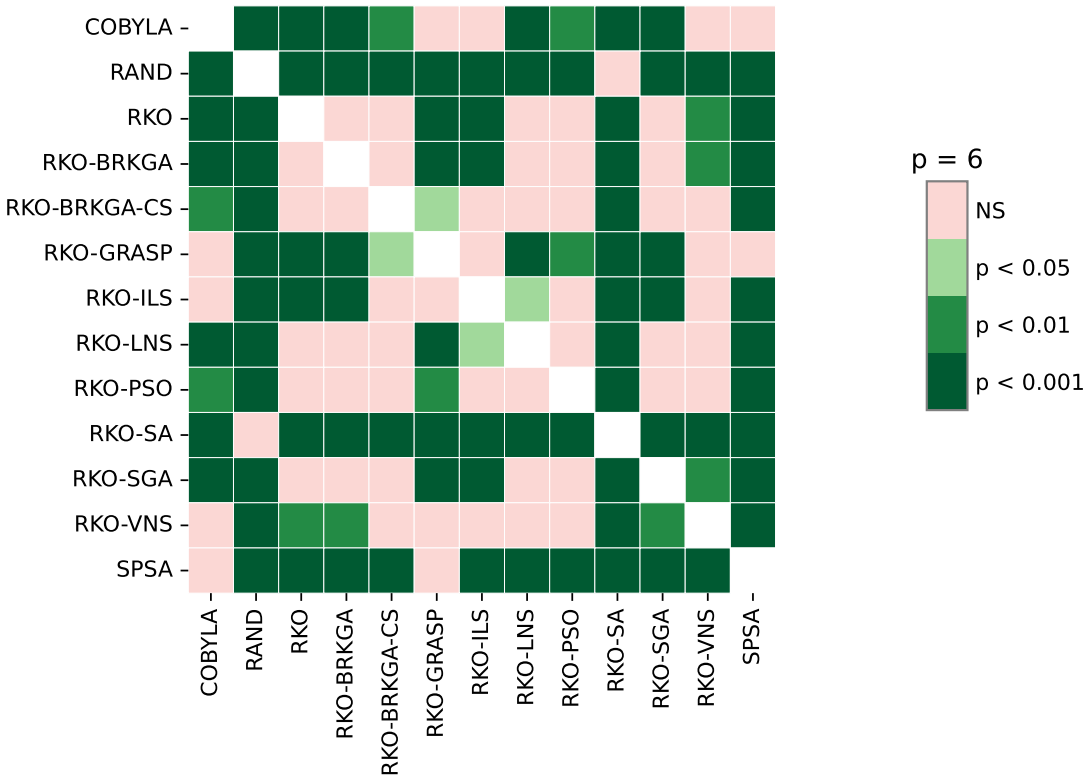


Figure 3.4.: The results of the Nemenyi test for QAOA parameter optimization ($p = 6$).

RKO-LNS improve the most, and are the best-performing methods overall, in terms of the median p_{opt}.

In terms of solved instances, the best methods are RKO and RKO-BRKGA. They now solve 84 instances, ten more than for $p = 3$. Figure 3.6 again depicts the solved instances for $p = 6$ against the number of decoder evaluations. SPSA now is clearly separated from RAND and RKO-SA. COBYLA and RKO-GRASP lag behind the larger group of methods on top. However, in contrast to $p = 3$, the curves do not flatten out near the end, indicating that most methods could still have improved with a higher evaluation budget. RKO-BRKGA-CS no longer stands out as it did for $p = 3$, which could be explained by the large difference in the hyperparameters selected for both settings (See Table A.1 in Appendix A).

Figure 3.7 shows the scaling behavior of RKO for this task for growing problem sizes

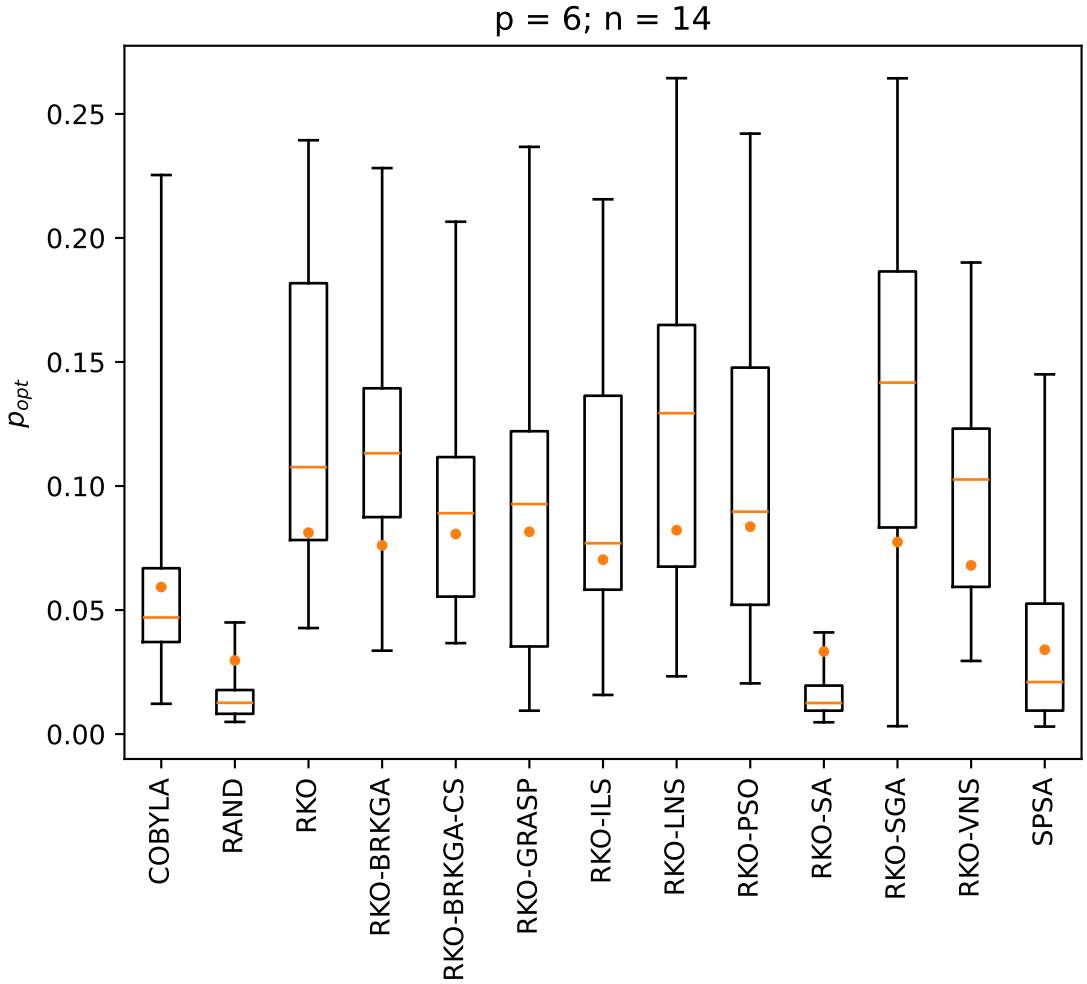


Figure 3.5.: Probability of measuring the optimum solution p_{opt} for each method, aggregated over problem instances of size $n = 14$ ($p = 6$). The boxes indicate the interquartile range. The whiskers cover the whole range of observed values. The orange line indicates the median. The orange dot indicates the median for ($p = 3$)

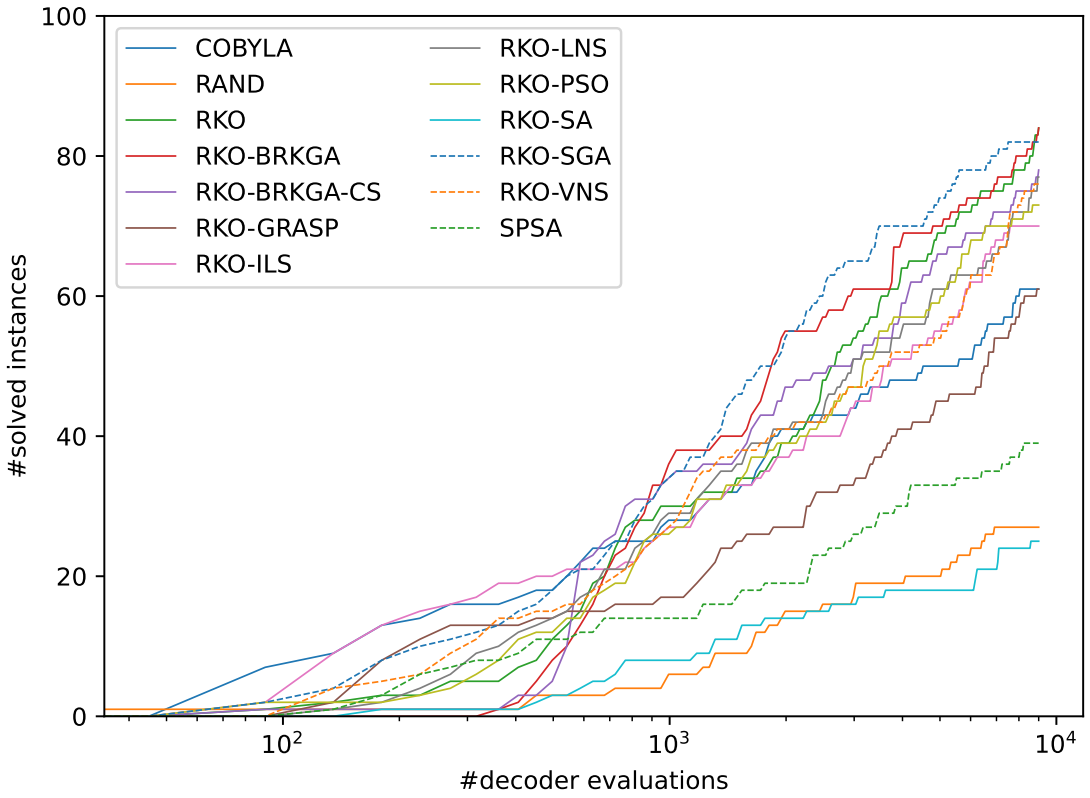


Figure 3.6.: The number of "solved" instances ($p_{opt} > 0.1$) for $p = 6$ plotted against the number of decoder evaluations on a logarithmic scale.

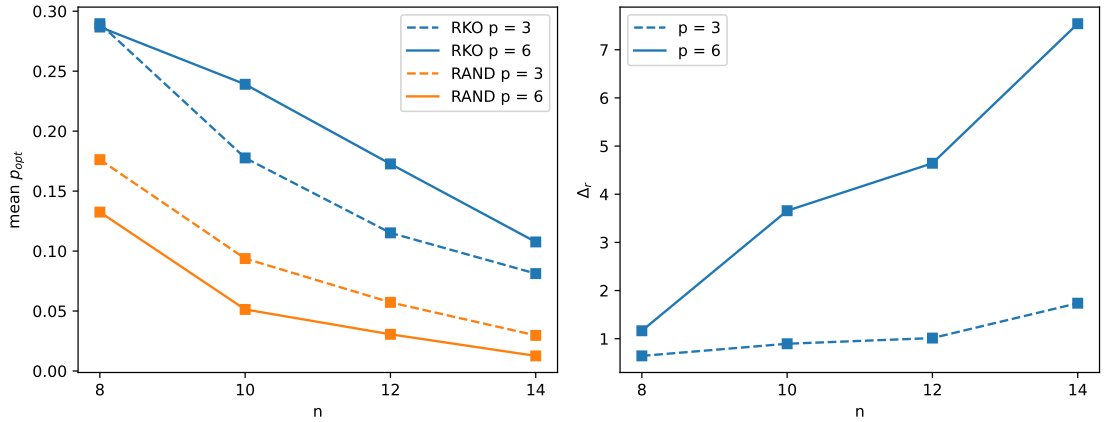


Figure 3.7.: Scaling of RKO compared to RAND with increasing problem size for $p = 3$ and $p = 6$. Absolute values (left) and relative difference $\Delta_r = (RKO - RAND)/RAND$ (right)

for both $p = 3$ and $p = 6$. As expected, the average p_{opt} achieved by RKO decreases with increasing problem sizes. At the same time, the effectiveness of random sampling decreases as well. The graph on the right displays the relative difference between the performance of RKO and RAND, showing that the advantage of RKO increases for growing problem sizes. The effect is stronger for $p = 6$, since RKO benefits from the additional free parameters, while the performance of RAND decreases. This, together with the previous results, indicates that the metaheuristics implemented in the RKO framework are able to effectively navigate the cost landscapes associated with the QAOA.

3.8. Discussion

Contrary to the results by Chaves. et al. [14], the full RKO optimizer does not converge significantly quicker than the individual metaheuristics. In the QAOA setting, the diversity provided by multiple different metaheuristics seems to be less beneficial than in the combinatorial optimization setting. Since the RKO framework utilizes a variety of different mechanisms, it is difficult to determine which components contribute the most to its performance on this task. However, the fact that RKO-SA is ineffective, while the remaining metaheuristics perform similarly, could hint at the importance of the local search methods. The RVND local search is frequently invoked by all metaheuristics, except RKO-SA. For RKO-SA, RVND is only invoked after SA_{max} iterations, which was

set to $SA_{max} = 465$ and $SA_{max} = 496$ for $p = 3$ and $p = 6$ respectively. In addition, since the evaluation budget for each individual thread of the metaheuristic is 1000, this choice of SA_{max} also causes the algorithm to rarely reduce the temperature. This is counter to the idea behind SA to model a continuous cooling process of a physical system [35]. Under these conditions, the RKO-SA heuristic seems to revert to random sampling. While this provides some insight into the performance of the RKO framework, that such a high value for SA_{max} was chosen for both settings could indicate that the number of trials for hyperparameter optimization was insufficient. Systematically testing different hyperparameter sets for RKO-SA could provide insight into whether this interpretation is correct, or whether the implementation of RKO-SA is inherently ill-suited for this task.

The poor performance of SPSA is further evidence of suboptimal hyperparameter choices. Our observations are in line with other results found in literature, which could indicate that SPSA is not well suited for the QAOA [10, 30, 48] in general. However, Sung et al. [62] report more positive results and argue that SPSA often performs poorly due to lacking hyperparameter optimization. Given our observation for RKO-SA above, this is likely also the case here.

Finally, the results for $p = 6$ show that not all metaheuristics scale equally well with increasing size of the key vectors. However, there is no clear advantage of one paradigm over another. For example, RKO-SGA and RKO-BRKGA-CS are both genetic algorithms, but the former scales significantly better than the latter. Similarly, RKO-LNS and RKO-ILS are similar in approach, but perform very differently for $p = 6$. Again, the variety of mechanisms implemented in the framework makes it difficult to pinpoint any causal relationships.

In summary, the results of this section show, that the RKO framework is applicable to the QAOA, surpassing the commonly chosen COBYLA optimizer in both our test settings. This is despite the fact that the search methods are not implemented with continuous optimization in mind, as noted in Section 2.4. This highlights the flexibility of metaheuristics in general and the RKO paradigm in particular. In the next section, we will make use of this flexibility to modify the standard QAOA ansatz with additional operators.

4. RKO for Operator Selection

There exist variations of the basic QAOA that utilize a different or extended set of operators in the ansatz. For example, Hadfield et al. [29] suggest using problem-specific mixing operators that restrict the evolution to a feasible subspace. Zhu et al. [69] propose a method where the ansatz is built layer by layer, each time selecting an operator with favorable gradient properties. In this section, we propose and evaluate a method for encoding the structure of the ansatz in a random key vector, in addition to the angle parameters. Ideally, the classical optimizer could then discover an effective ansatz at runtime. We first present a decoder design that allows a selection of operators in each layer. Then, we describe two additional operators and motivate their selection before comparing the performance of this design to the results of Section 3.

4.1. Structure Decoder

We replace the QAOA ansatz with a more general layered ansatz.

$$|\psi(\theta)\rangle = U_d(\theta_d)U_{d-1}(\theta_{d-1})\cdots U_2(\theta_2)U_1(\theta_1)\,|+\rangle^{\otimes n} \tag{4.1}$$

Here, the operations U_i are not fixed but can be chosen from an operator pool $P = (P_0, \ldots, P_{m-1})$ of m operators. For simplicity, we require each operator to use one variational parameter θ_i. We can then define a decoder that decodes a variational ansatz and corresponding parameters θ from a random key vector $\chi = (\chi_1, \ldots, \chi_d)$ with

$$j = \lfloor m\chi_i \rfloor \qquad U_i = P_j \qquad \theta_i = 2\pi(\chi_i - \frac{j}{m}) \tag{4.2}$$

Note that a single key χ_i encodes the operator together with the corresponding parameter. This results in intuitive semantics for the random key optimization. For example, a swap operation of two keys simply translates to swapping two operators.

4.2. Counterdiabatic Driving

In Section 2.6, we noted that the QAA requires sufficient evolution times T to avoid transitions to higher energy states. In principle, it is possible to suppress these transitions by adding an additional term $\mathcal{A}$, the so-called adiabatic gauge potential, to the

time-dependent Hamiltonian [16].

$$H'(t) = H(t) + \mathcal{A} \tag{4.3}$$

This process is referred to as counterdiabatic driving. In general, computing the adiabatic gauge potential is as difficult as solving the problem itself, because it requires diagonalizing H [16]. However, it can be approximated using a nested commutator expansion. Let $[A, B] = AB - BA$ denote the commutator of two square matrices of matching dimension. Then

$$\mathcal{A} = \lim_{l \to \infty} i \sum_{k=1}^{l} \alpha_k \underbrace{[H, [H, [H,}_{2k-1} \partial_t H]]] \tag{4.4}$$

for some coefficients $\alpha_1, \ldots, \alpha_k$. Chandarana et al. [13] show that by inserting terms from low order approximations ($l = 2$) of $\mathcal{A}$ into the standard QAOA ansatz, the algorithm can achieve faster convergence and higher approximation ratios. The α coefficients are set as variational parameters. In a similar vein, Wurtz and Love [66] show that the standard QAOA can already leverage counterdiabatic driving, since for appropriate angle parameters θ, the error of the lowest order Suzuki-Trotter decomposition matches low order approximations of $\mathcal{A}$. In particular, the error contains a component of the form $i\alpha[H_C, H_B]$. Therefore, we extend the operator pool of the algorithm by an operator U_{CD}, which should allow it to explicitly make use of counterdiabatic driving.

$$U_{CD}(\alpha) = e^{\alpha[H_C, H_B]} \tag{4.5}$$

4.3. Phantom Edges

As discussed in Section 2.6, locality can limit the performance of QAOA. Langfitt et al. [38] propose introducing additional edges into the problem graph to decrease the average distance between nodes. The contribution of these "phantom edges" to the problem Hamiltonian is controlled by a variational parameter α. Their approach is most successful if the phantom edges are chosen to introduce triangles into the graph. Formally, for a graph $G = (V, E)$, E' are the phantom edges with

$$E' = (u, v) \notin E | \exists w \in V . (u, w) \in E \wedge (v, w) \in E \tag{4.6}$$

We mimic this approach by introducing the operator U_Δ into the operator pool with

$$H_\Delta = \sum_{(u,v) \in E'} \sigma_u^z \sigma_v^z \qquad\qquad U_\Delta(\alpha) = e^{-i\alpha H_\Delta} \tag{4.7}$$

4.4. Results

The experiment is conducted identically to Section 3, with only the decoder exchanged for the structure decoder. The operator pool P consists of the two standard QAOA operators defined over H_C and H_B, the counterdiabatic operator U_{CD}, and the triangle operator U_Δ. We set $d = 6$, which results in the same amount of layers as $p = 3$ in the previous experiment. The hyperparameters used are presented in Table A.2.

The Friedman test rejects the null hypothesis with $p = 6.25e - 188$. Figure 4.1 shows the results of the Nemenyi test. A large block of methods between RKO and RKO-LNS can not be distinguished. In addition, RKO-PSO is no longer distinguishable from RAND, meaning it fails to effectively navigate the cost landscape. The same is true for RKO-SA, providing further evidence that it does not work well in this setting. In contrast to the $p = 3$ setting in the previous experiment, SPSA can be distinguished from RAND, but as Figure 4.2 shows, this is because SPSA performs significantly worse than RAND, instead of better. No clear pattern is visible for the remaining methods. In general, we observe a large drop in the median p_{opt} for all methods.

The same loss in quality can be observed in the number of solved instances in Figure 4.3. COBYLA and RKO-BRKGA now solve the most instances, but the number has reduced from 74 in the previous $p = 3$ case to only 50. However, the slope has also decreased significantly, and the curves for most methods do not flatten out, so further improvements might be possible using additional iterations.

Since RKO-BRKGA is the most successful metaheuristic, we also investigate whether it exhibits a learning behavior where a trend towards a particular selection of operators is visible. For this purpose, we plot the operator prevalence for one run of RKO-BRKGA in Figure 4.4. The left column corresponds to the hardest instance of size $n = 8$, and the right column to the easiest instance of size $n = 14$, as determined in Section 3. To reduce visual noise, the graphs are moving averages of the respective values. The first row shows the number of occurrences for a given operator overall. The remaining rows show the operator prevalence at the positions of U_1, U_2, and U_3, respectively. Firstly, all plots show a strongly periodic behavior. During phases where the population of the genetic algorithm is refilled with mutants, the curves overlap since all operators are equally likely. During phases of local search and crossover, the operator selection strongly tends towards a certain distribution. For the first row on the left, H_B and U_{CD} are most prevalent during these phases. On the right, for $n = 14$, the distribution is less clear, but U_Δ rises above the other operators for extended intervals. The second row provides a good sanity check for the algorithm. Since the

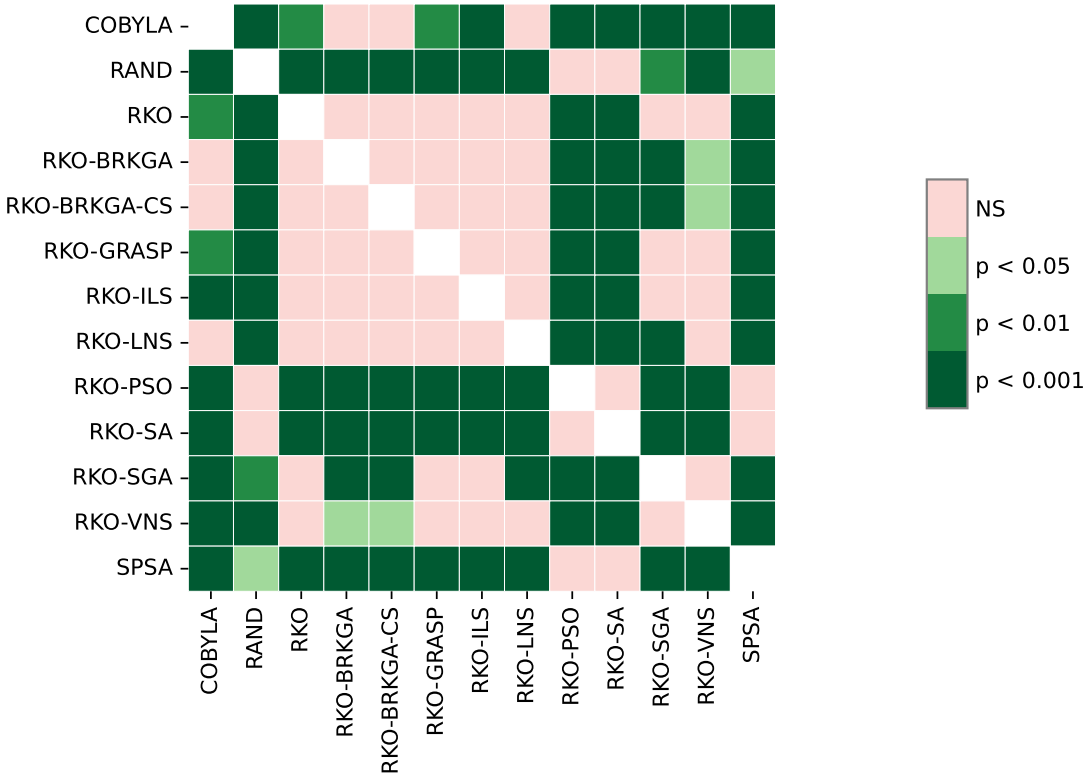


Figure 4.1.: The results of the Nemenyi test for the structure decoder.

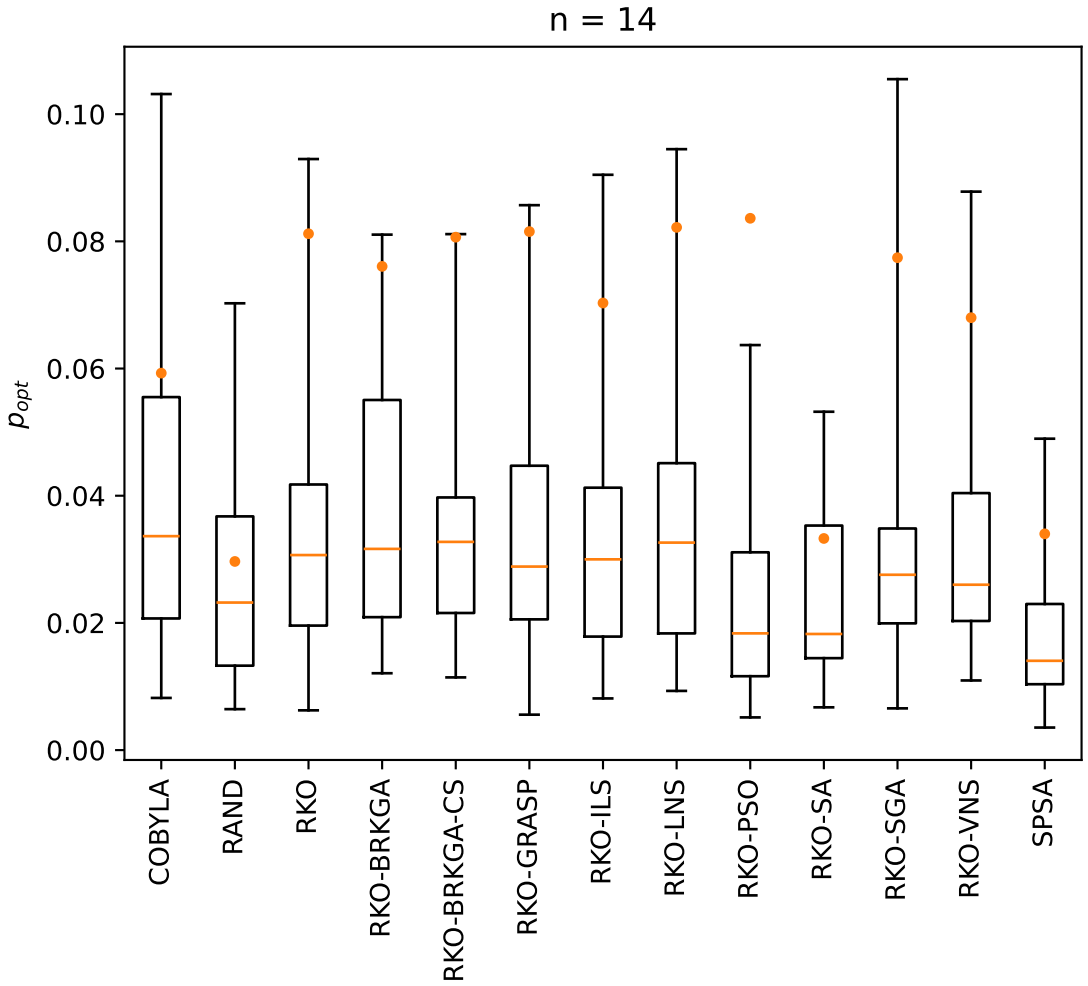


Figure 4.2.: Probability of measuring the optimum solution p_{opt} for each method for the structure decoder, aggregated over problem instances of size $n = 14$. The boxes indicate the interquartile range. The whiskers cover the whole range of observed values. The orange line indicates the median.

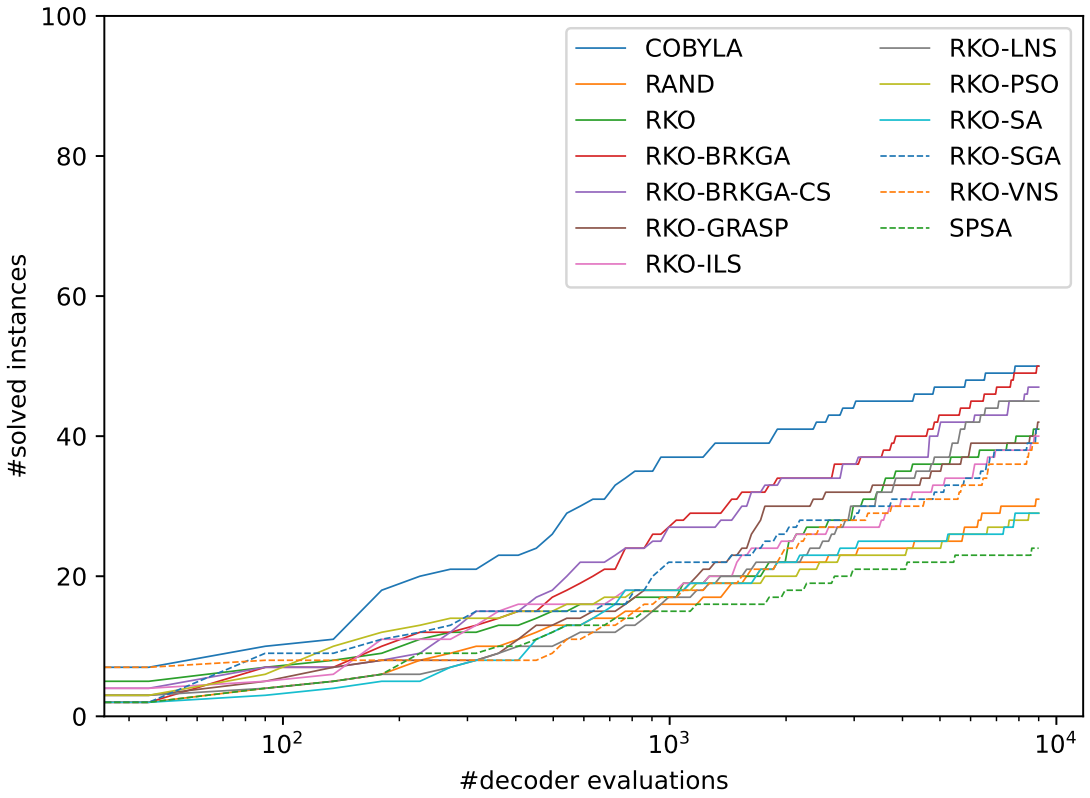


Figure 4.3.: The number of "solved" instances ($p_{opt} > 0.1$) for the structure decoder plotted against the number of decoder evaluations on a logarithmic scale.

initial state $|+\rangle^{\otimes n}$ is an eigenstate of H_B, the mixing operator has no effect in the first position. Therefore, key vectors with this feature tend to perform poorly and are subsequently suppressed by the selection mechanism. U_{CD} is most prevalent in the first position for both instances, replacing H_C in the standard QAOA ansatz. H_B dominates the second position, highlighting the importance of the mixing operator. In the third row, all four operators experience peaks at different times. No clear preference is visible.

4.5. Discussion

In summary, the analysis of operator prevalence shows that the RKO-BRKGA metaheuristic develops clear preferences for certain ansatz configurations, which differ from the standard QAOA ansatz. Firstly, differences in the operator selection between instances could indicate, that the algorithm can adapt the ansatz to different situations. Secondly, these configurations could hint at ansatz structures that are indeed more suited to the task than the standard ansatz, but the heuristics fail to fine-tune the ansatz parameters. This might be related to the origin of the RKO framework in combinatorial optimization instead of continuous optimization. One way to mitigate this would be to fix the structure after a certain number of iterations. The task would then revert to standard VQA parameter tuning.

Overall, the introduction of additional operators into the ansatz via the structure decoder has not improved performance in any metric. On the contrary, all methods achieve a lower mean p_{opt} and solve fewer instances. This is likely because the increased complexity of the cost landscape negates any potential benefits the operators could provide, at least in the setting of this particular experiment. We have not conducted experiments with variations in the circuit depth or the overall evaluation budget. Changing these values could potentially produce a different picture, akin to the improvements seen in Section 3. Further investigating the scaling behavior and potential refinements of this approach is warranted.

In the next section, we will move away from the MaxCut problem and instead attempt to use the RKO paradigm to address COPs that are more difficult to encode on quantum hardware.

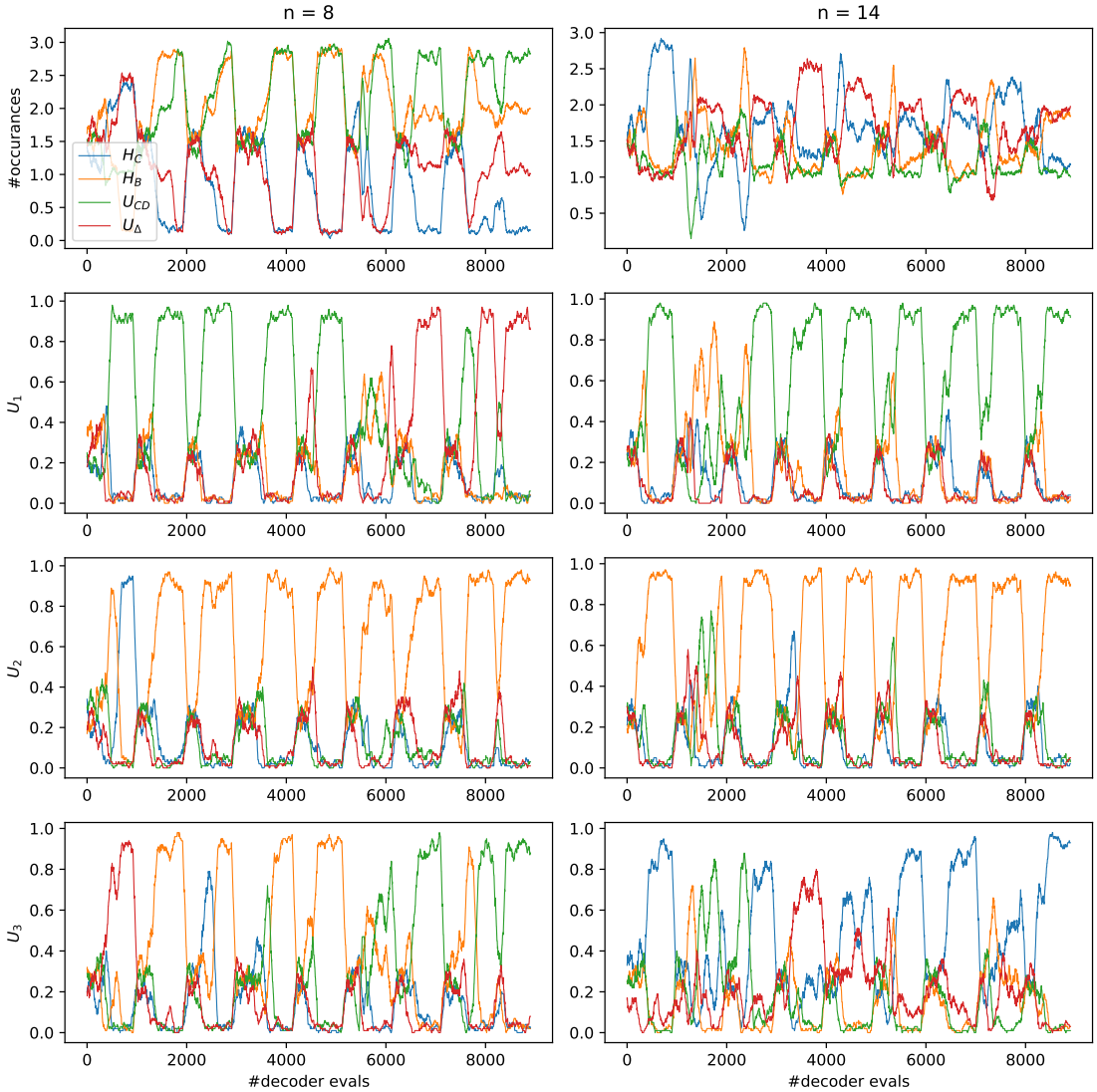


Figure 4.4.: Operator prevalence against decoder evaluations for two exemplary instances of size $n = 8$ (left) and $n = 14$ (right). For readability, all graphs are moving averages with a window size of 100 points. The first row shows the overall prevalence of an operator at any position. Rows two to four show the prevalence of operators selected for U_1, U_2, and U_3 respectively.

5. RKO for Non-Native Poblems

In Section 2.3, we noted that the reduction of the TSP to an Ising model incurs a quadratic overhead in the number of variables, which limits the size of instances that can be represented on NISQ devices. Wurtz et al. [67] refer to problems that suffer from this issue as non-native problems and propose to use different classical post-processing routines to address them. In this section, we present such a routine inspired by RKO, which uses the expectation values of qubit correlations to efficiently decode candidate solutions for TSP instances while only using a linear amount of qubits or less.

5.1. Polynomial Compression

In a typical application of a VQA for combinatorial optimization, each variable in the original problem is represented by one qubit on the device. Sciorilli et al. [60] propose to use Pauli correlations between qubits instead. Since the number of correlations that can be constructed between n qubits rises polynomially in n, this reduces the number of qubits necessary to represent a given problem. In addition, they show that this approach can improve ansatz trainability by mitigating barren plateaus in the cost landscape [60].

Formally, for a problem defined over spins $s = (s_1, \ldots, s_m)$, choose a set of observables $\Pi = \{\Pi_i | i \in [m]\}$ with

$$\Pi \subseteq \{F_1 \otimes \cdots \otimes F_k | F \in \{\mathbb{1}, \sigma^x, \sigma^y, \sigma^z\}\} / \{\mathbb{1}^{\otimes k}\} \tag{5.1}$$

where k is the order of the polynomial compression. The spins s_i are then determined by the signs of the corresponding expectation value of Π_i on the output state of a variational ansatz $|\psi(\theta)\rangle$.

$$s_i := sgn(\langle\psi(\theta)| \Pi_i |\psi(\theta)\rangle) \tag{5.2}$$

Note that, since only measurements of commuting observables can be performed in parallel, multiple measurement settings are needed if Π contains observables that do not mutually commute [60].

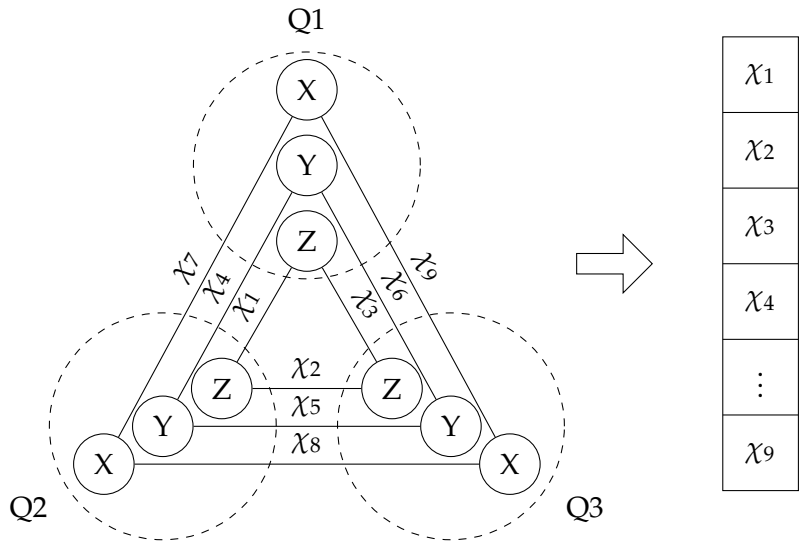


Figure 5.1.: The polynomial compression scheme. Figure adapted from Sciorilli et al. [60]. Instead of only considering the signs, we use the full expectation values to construct a random key vector χ

.

For this experiment, we choose Π to contain all $\sigma^x\sigma^x$, $\sigma^y\sigma^y$, and $\sigma^z\sigma^z$ correlations for all pairs of qubits. The compression scheme for this choice is illustrated in Figure 5.1. For n qubits, this yields m correlations where

$$m = \frac{3n(n-1)}{2} \tag{5.3}$$

5.2. Permutation Decoder

For decoding solutions to the TSP, we modify this compression method. Instead of using the signs of the expectation values, we interpret the expectation values as elements of a random key vector (Figure 5.1). We then use the sorting decoder proposed by Bean [5] to obtain a valid permutation p

$$p = argsort(\langle\Pi_1\rangle, \ldots, \langle\Pi_m\rangle) \tag{5.4}$$

where $\langle\Pi_i\rangle$ is shorthand for $\langle\psi(\theta)|\,\Pi_i\,|\psi(\theta)\rangle$, and *argsort* returns the permutation induced by sorting the values in ascending order. Technically, the expectation values $\langle\Pi_1\rangle$ are in the interval $[-1, 1]$ instead of $[0, 1)$. For the purposes of the permutation

decoder, this is irrelevant. For general decoding strategies, this could be corrected with a simple transformation.

Note that it is unclear whether every permutation can be represented by this encoding. Sciorilli et al. provide a sufficient condition for a compression scheme to be able to represent all possible bit strings [60]. However, this proof is not transferable to the modified compression scheme, and we have not attempted to devise a similar proof. Therefore, it is possible that for a given problem instance, no quantum state exists that corresponds to the optimal solution.

5.3. Variational Ansatz

Due to the polynomial compression, the standard QAOA ansatz can not be used in this setting. Sciorilli et al. instead use a brickwork ansatz as depicted in Figure 5.2 [60]. Here, every even layer consists of two-qubit Mølmer-Sørensen (MS) gates with three variational parameters each, while every odd layer consists of single-qubit rotation gates with one parameter each. The axis of the rotation gates cycles through X, Y, and Z between layers, and the MS gates alternately connect even or odd neighbors. We also chose this ansatz for our experiment, since it has been effective in the experiments by Sciorilli et al. Generally, the choice of ansatz may be another limiting factor for the performance of the algorithm. Even if a state exists that would decode into the optimal solution, it might not be reachable with a given ansatz.

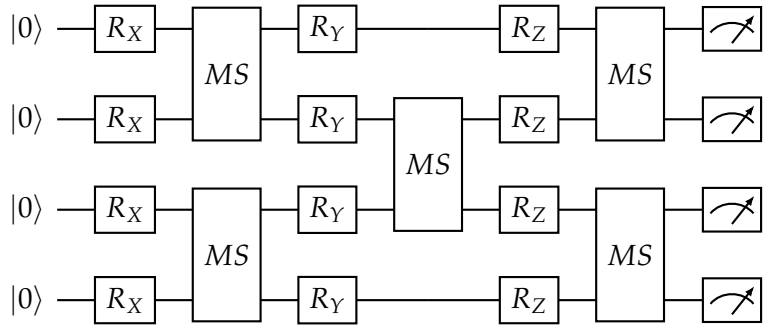


Figure 5.2.: Brickwork ansatz used by Sciorilli et al.[60] with four qubits and six layers.

5.4. Experimental Setup

Due to the computational complexity of the simulation, we limit experiments to at most 14 qubits. According to Equation 5.3, 14 qubits provide 273 correlations in our chosen encoding. By fixing the starting point of the route, the proposed decoder can then decode solutions for TSP instances with up to 274 cities. As test instances, we use the 18 TSP instances of the TSPLIB [56] problem library that fulfill this requirement and contain a precomputed solution. We use an ansatz with six layers. For the smallest test instance with 16 cities on four qubits, this corresponds to 27 free parameters. The largest instance, with 225 cities on 13 qubits, uses an ansatz with 93 free parameters. Table B.1 lists all problem instances and the corresponding ansatz specifications. We perform hyperparameter optimization using three medium-sized instances with $n = 76$, $n = 96$, and $n = 100$. The process is as described in Section 3.3. The remaining 15 instances form the test set. The chosen hyperparameters are listed in Table A.3.

We also need to deviate from the methodology of the previous sections with regard to the cost function. The CVaR cost function is only applicable when dealing with a distribution of solutions. Previously, a certain set of parameters for the ansatz would produce multiple solution candidates by repeated measurements. Since we now require sets of expectation values instead of bit strings, the distribution produced by the circuit itself encodes a single solution candidate. Therefore, we use the approximation ratio (AR) instead, computed using the solutions provided by the TSPLIB dataset. The AR of a solution candidate s is defined as

$$AR = \frac{C(s)}{C(s_{min})} \tag{5.5}$$

where C is the cost function of the optimization problem, and s_{min} is the optimal solution. An AR of 1 indicates that the problem was solved optimally. Since the TSP is a minimization problem, $AR >= 1$, and lower values are better.

The methods used to train the ansatz parameters are the same as before. To additionally capture the difference between directly sampling the solution space and sampling mediated through the variational ansatz, we compare the results to the method DIRRAND, which simply produces uniformly distributed random permutations.

5.5. Results

The Friedman test rejects the null hypothesis with $p = 1.53e - 33$. Figure 5.3 shows the results of the Nemenyi test. Only a few methods exhibit significant differences. There

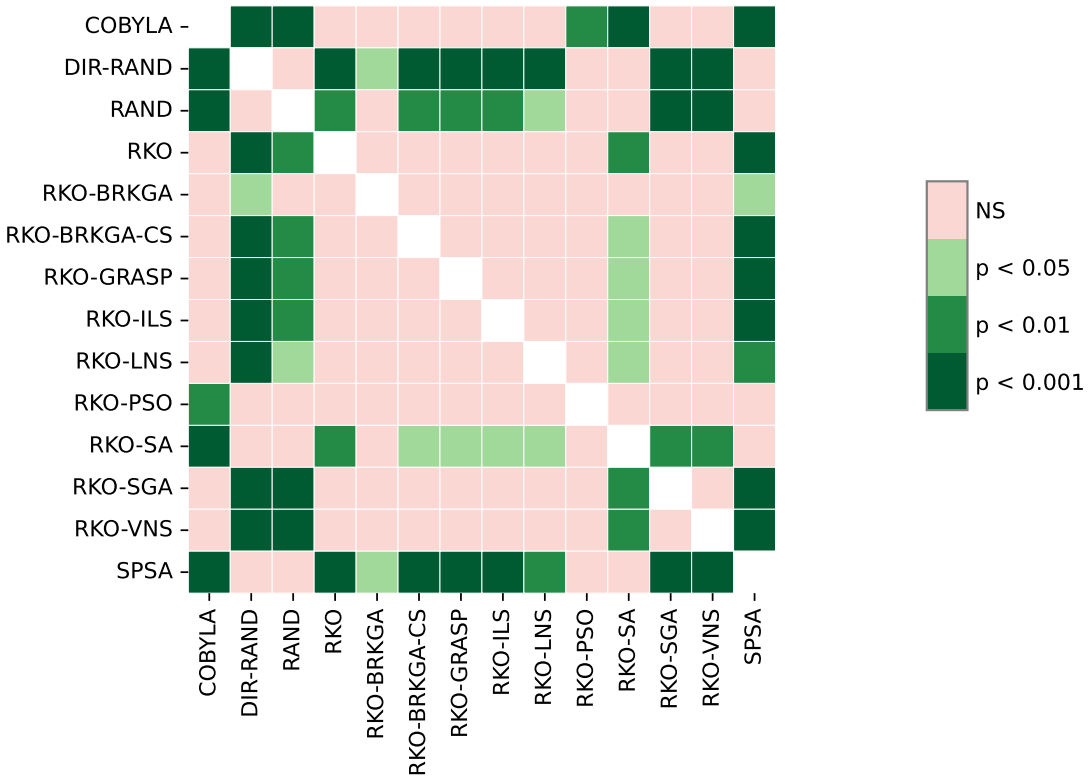


Figure 5.3.: The results of the Nemenyi test for the TSP using polynomial qubit compression and RKO decoding.

are four clear lines, as DIR-RAND, RAND, RKO-SA, and SPSA are mostly distinguishable from the other methods. Figure 5.4 plots the AR achieved by each method for the individual instances. The dashed line marks $AR = 1$, meaning that an instance would be solved. No method achieves the optimal solution for any instance. For some pairs of methods, a difference in performance on one instance correlates to a difference in performance for other instances. However, this relationship does not hold for all pairs of methods, explaining the results of the Nemenyi test. This is most visible for the two instances with $n > 200$. Larger test sets could help to draw out more significant differences between methods.

Figure 5.5 shows the convergence behavior of all methods by plotting the mean AR against the number of decoder evaluations. Only COBYLA and DIR-RAND appear to flatten out near the end. The remaining methods could likely have improved with

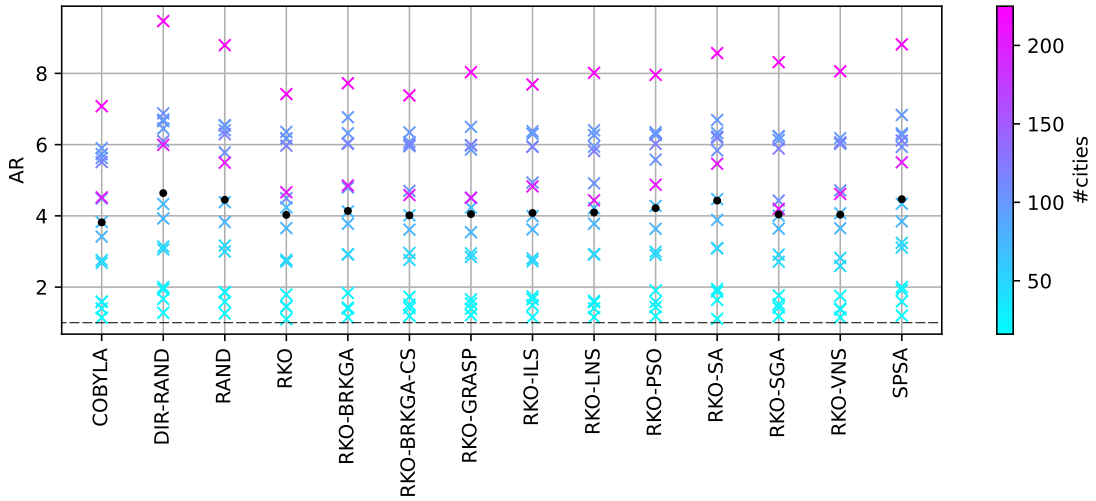


Figure 5.4.: AR achieved by all methods for individual TSP instances. The size of the instance is indicated by the color map. The black dot marks the mean AR achieved. The dashed line marks $AR = 1$. No method reaches this value for any instance.

additional decoder evaluations. RKO-SA and SPSA are again closely grouped with RAND. COBYLA is continuously ahead, but due to the results of the Nemenyi test, this is not a significant result.

Finally, the histograms in Figure 5.6 show the sampling distribution of cost function values between RAND, DIR-RAND, and COBYLA for problem instances 2 and 18. In both cases, COBYLA is able to sample better values than both forms of random sampling. This shows that it is principally able to navigate the cost landscape. For instance 18, we additionally observe that the sampling distribution of RAND is significantly shifted against the distribution of DIR-RAND. Since the ansatz does not contain any problem-specific information, the fact that it is shifted in the positive direction is a coincidence. Rearranging the qubits could produce different results. In general, this shows that the chosen ansatz biases the algorithm towards specific states, which could prevent the overall optimization routine from reaching more desirable states.

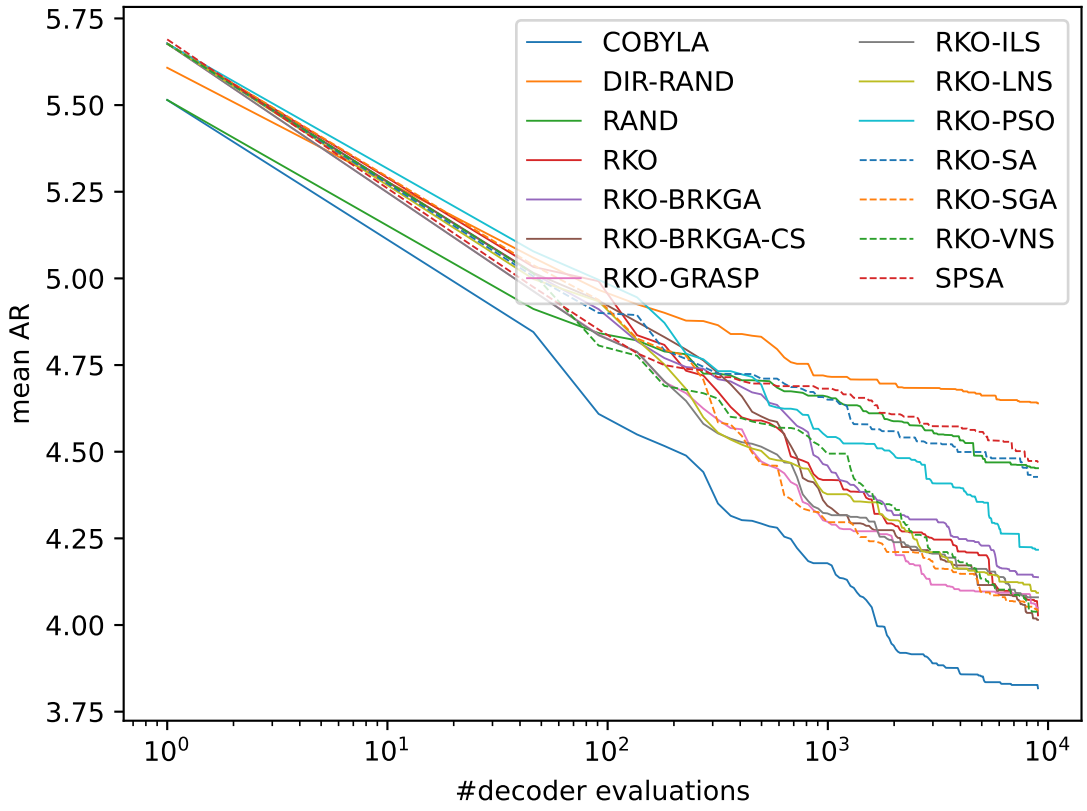


Figure 5.5.: The mean approximation ratios achieved by the methods against the number of decoder evaluations on a logarithmic scale.

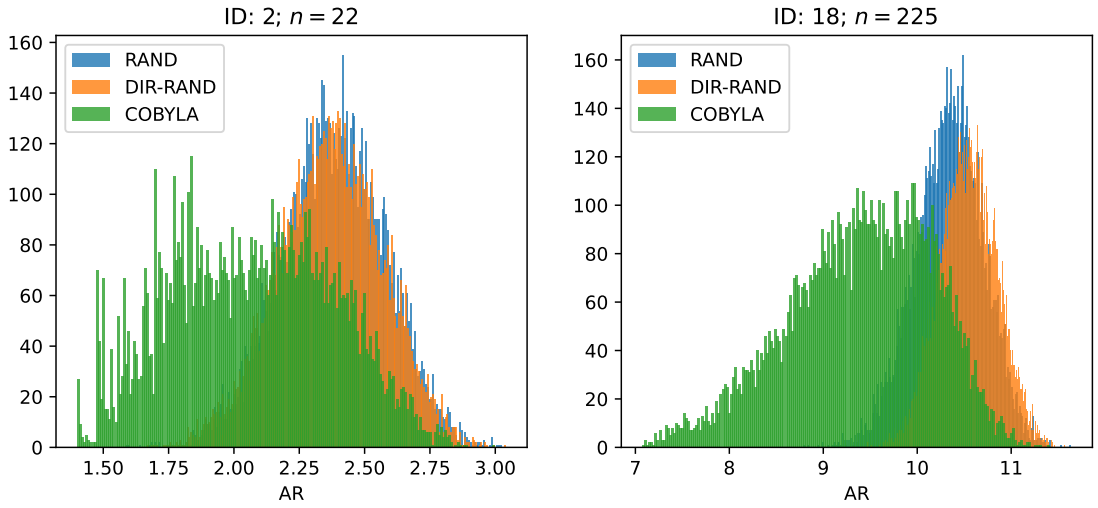


Figure 5.6.: The sampling distribution of cost function values between RAND, DIR-RAND, and COBYLA for problem instances 2 and 18, respectively. For instance, 18, the sampling mediated by the variational circuit is clearly distinguished from the direct sampling.

5.6. Discussion

We have shown that the proposed method introduces a bias compared to uniform random sampling, potentially limiting performance. Analytically investigating the properties of the solution encoding and the ansatz could provide insights into theoretical limits and avenues for improvement. Progress in solving the TSP using this method could directly yield improvements for other non-native problems by adapting the decoder accordingly.

Overall, the proposed method does not produce satisfactory results for any TSP instance tested, but we have shown that the classical optimizers can significantly shift the sampling distribution towards better results compared to random sampling. However, this is only minor evidence for the viability of the approach, and further numerical experiments are necessary. Apart from RKO-SA and SPSA, we could not observe significant differences between the methods, indicating that no paradigm handles the task particularly well. The poor performance of RKO-SA and SPSA is a commonality between all experiments of this thesis. We will address this and other conclusions in the next section.

6. Conclusions

In this thesis, we have conducted experiments on the performance of the RKO framework by Chaves et al. [14] for tasks related to VQAs. We have also proposed two RKO inspired modifications to standard VQA approaches. We could show that the RKO framework is suitable for the basic task of QAOA parameter optimization, but we could not produce similarly positive results for the modified approaches.

There are some commonalities between the results of all three experiments. Firstly, RKO-SA and SPSA consistently struggle to produce competitive results compared to the other methods. For SPSA, we noted that Sung et al. [62] attribute this to lacking hyperparameter optimization. This could also be the case for RKO-SA, which would mean that both methods are significantly more difficult to parameterize effectively. It could also hint at more inherent issues of these methods when applied to the tasks in this thesis. Secondly, we could not identify clear patterns in the performance of the other metaheuristics. No metaheuristic consistently outperforms any other method, and the RKO framework does not seem to reap the same benefit from metaheuristic diversity as reported by Chaves et al. [14]. This, together with the suboptimal hyperparameter selection, bars us from drawing more significant conclusions.

In general, this thesis provides a broad overview of the intersection between RKO and VQAs, at the expense of a more in depth analysis. Both experiments in Section 4 and Section 5 hint at potential issues of the respective approaches, which warrant further investigation. Additional experiments into the scaling behavior of the tested methods could also provide valuable insight. The code and data used in this thesis are available at https://github.com/AlexanderTreml/QRKO-Thesis

A. Hyperparameters

$p = 3$				$p = 6$			
RKO-BRKGA		RKO-SA		RKO-BRKGA		RKO-SA	
p	382	SA_{max}	465	p	322	sA_{max}	496
p_e	0.0137	α	0.7678	p_e	0.0166	α	0.2276
p_m	0.5055	β_{min}	0.0756	p_m	0.7427	β_{min}	0.2309
ρ_e	0.7674	β_{max}	0.0070	ρ_e	0.7377	β_{max}	0.7414
		T_0	4138542			T_0	8832280
RKO-GRASP		RKO-ILS		RKO-GRASP		RKO-ILS	
α	0.0160	β_{min}	0.3903	α	0.1300	β_{min}	0.1445
h_s	0.5830	β_{max}	0.4161	h_s	0.9097	β_{max}	0.7111
h_e	0.0076			h_e	0.0021		
RKO-VNS		RKO-PSO		RKO-VNS		RKO-PSO	
k_{max}	96	p	177	k_{max}	86	p	5
β_{min}	0.8021	c_1	2.3558	β_{min}	0.3760	c_1	9.9256
		c_2	9.4573			c_2	3.6392
		w	0.1226			w	1.6666
RKO-SGA		RKO-LNS		RKO-SGA		RKO-LNS	
p	174	β_{min}	0.3331	p	4	β_{min}	0.8226
p_c	0.0444	β_{max}	0.6793	p_c	0.5700	β_{max}	0.2154
μ	0.1683	T_0	6109826	μ	0.1321	T_0	6876865
		α	0.1114			α	0.4301
RKO-BRKGA-CS		COBYLA		RKO-BRKGA-CS		COBYLA	
p	65	*rhobeg*	0.1815	p	236	*rhobeg*	0.5975
p_e	0.0530	*tol*	0.0006	p_e	0.0534	*tol*	0.0009
p_m	0.2931			p_m	0.6138		
ρ_e	0.7217			ρ_e	0.9830		
SPSA				SPSA			
α	0.7663			α	0.8225		
γ	0.1281			γ	0.1514		
c	0.0406			c	0.0585		
A	908.9013			A	538.6038		
a	0.8063			a	0.8113		
patience	340			*patience*	748		

Table A.1.: Hyperparameters used in Section 3

RKO-BRKGA		RKO-SA	
p	222	SA_{max}	326
p_e	0.0687	α	0.7006
p_m	0.6517	β_{min}	0.7893
ρ_e	0.8815	β_{max}	0.2054
		T_0	3217348
RKO-GRASP		RKO-ILS	
α	0.8428	β_{min}	0.0134
h_s	0.5686	β_{max}	0.8352
h_e	0.0536		
RKO-VNS		RKO-PSO	
$kMax$	78	$PSize$	260
β_{min}	0.2505	c_1	5.3649
		c_2	5.4446
		w	5.6354
RKO-SGA		RKO-LNS	
p	102	β_{min}	0.8226
p_c	0.6425	β_{max}	0.2154
μ	0.1913	T_0	6876865
		α	0.4301
RKO-BRKGA-CS		COBYLA	
p	230	$rhobeg$	0.2357
p_e	0.0047	tol	0.0006
p_m	0.4092		
ρ_e	0.9912		
SPSA			
α	0.6058		
γ	0.1420		
c	0.2358		
A	732.1952		
a	0.4288		
$patience$	645		

Table A.2.: Hyperparameters used in Section 4

RKO-BRKGA		RKO-SA	
p	20	SA_{max}	613
p_e	0.0873	α	0.1593
p_m	0.0998	β_{min}	0.1019
ρ_e	0.8285	β_{max}	0.0187
		T_0	6831366
RKO-GRASP		RKO-ILS	
α	0.5165	β_{min}	0.0993
h_s	0.1288	β_{max}	0.3098
h_e	0.0613		
RKO-VNS		RKO-PSO	
k_{max}	89	p	37
β_{min}	0.3302	c_1	0.5054
		c_2	5.3294
		w	5.0747
RKO-SGA		RKO-LNS	
p	29	β_{min}	0.8995
p_c	0.0708	β_{max}	0.6388
μ	0.0543	T_0	13409
		α	0.5312
RKO-BRKGA-CS		COBYLA	
p	120	*rhobeg*	0.2413
p_e	0.6118	*tol*	0.0005
p_m	0.4189		
ρ_e	0.9008		
SPSA			
α	0.9372		
γ	0.1486		
c	0.3758		
A	988.2278		
a	0.5129		
patience	352		

Table A.3.: Hyperparameters used in Section 5

B. TSP instances

ID	name	#cities	#qubits	#free params
1	ulysses16	16	4	27
2	ulysses22	22	5	33
3	gr24	24	5	33
4	fri26	26	5	33
5	gr48	48	7	48
6	eil51	51	7	48
7	st70	70	8	57
8	eil76	76	8	57
9	pr76	76	9	63
10	gr96	96	9	63
11	rd100	100	9	63
12	kroD100	100	9	63
13	kroA100	100	9	63
14	kroC100	100	9	63
15	lin105	105	9	63
16	gr120	120	10	72
17	gr202	202	13	93
18	tsp225	225	13	93

Table B.1.: The traveling salesman problem (TSP) instances used in Section 5. The instances marked in grey were used for hyperparameter optimization.

Abbreviations

BRKGA Biased Random Key Genetic Algorithm

COBYLA Constrained Optimization By Linear Approximation

COP combinatorial optimization problem

CS clustering search

CVaR conditional value-at-risk

GRASP Greedy Randomized Adaptive Search Procedure

ILS Iterated Local Search

LNS Large Neighborhood Search

MS Mølmer-Sørensen

NISQ noisy intermediate-scale quantum

PSO Particle Swarm Optimization

QAA Quantum Adiabatic Algorithm

QAOA Quantum Approximate Optimization Algorithm

QUBO quadratic unconstrained binary optimization

RKGA Random Key Genetic Algorithm

RKO random key optimization

RVND Random Variable Neighborhood Descent

SA Simulated Annealing

SPSA Simultaneous Perturbation Stochastic Approximation

TSP traveling salesman problem

UQBP unconstrained binary quadratic programming

VNS Variable Neighborhood Search

VQA variational quantum algorithm

w3R weighted three-regular graph

List of Figures

List of Tables

Bibliography

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. "Optuna: A Next-Generation Hyperparameter Optimization Framework." In: *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2623–2631.

[2] B. Augustino, M. Cain, E. Farhi, S. Gupta, S. Gutmann, D. Ranard, E. Tang, and K. Van Kirk. *Strategies for running the QAOA at hundreds of qubits*. 2024. DOI: 10.48550/ARXIV.2410.03015.

[3] F. Barahona. "On the computational complexity of Ising spin glass models." In: *Journal of Physics A: Mathematical and General* 15.10 (1982), p. 3241.

[4] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner. "Improving Variational Quantum Optimization using CVaR." In: (2019). DOI: 10.48550/ARXIV.1907.04769.

[5] J. C. Bean. "Genetic Algorithms and Random Keys for Sequencing and Optimization." In: *ORSA Journal on Computing* 6.2 (May 1994), pp. 154–160. ISSN: 2326-3245. DOI: 10.1287/ijoc.6.2.154.

[6] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. Di Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2018. DOI: 10.48550/ARXIV.1811.04968.

[7] S. Bolusani, M. Besançon, K. Bestuzheva, A. Chmiela, J. Dionísio, T. Donkiewicz, J. van Doornmalen, L. Eifler, M. Ghannam, A. Gleixner, C. Graczyk, K. Halbig, I. Hedtke, A. Hoen, C. Hojny, R. van der Hulst, D. Kamp, T. Koch, K. Kofler, J. Lentz, J. Manns, G. Mexi, Erik Mühmer, M. E. Pfetsch, F. Schlösser, F. Serrano,

Y. Shinano, M. Turner, S. Vigerske, D. Weninger, and L. Xu. *The SCIP Optimization Suite 9.0*. Technical Report. Optimization Online, Feb. 2024.

[8] M. Born and V. Fock. "Beweis des Adiabatensatzes." In: *Zeitschrift fuer Physik* 51.3–4 (Mar. 1928), pp. 165–180. ISSN: 1434-601X. DOI: 10.1007/bf01343193.

[9] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang. "Obstacles to Variational Quantum Optimization from Symmetry Protection." In: *Physical Review Letters* 125.26 (Dec. 2020), p. 260505. ISSN: 1079-7114. DOI: 10.1103/physrevlett.125.260505.

[10] Britant and A. Pathak. *Revisiting Majumdar-Ghosh spin chain model and Max-cut problem using variational quantum algorithms*. 2024. DOI: 10.48550/ARXIV.2404.18142.

[11] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles. "Variational quantum algorithms." In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9.

[12] Y. Chai, K. Jansen, S. Kühn, T. Schwägerl, and T. Stollenwerk. *Structure-inspired Ansatz and Warm Start of Variational Quantum Algorithms for Quadratic Unconstrained Binary Optimization Problems*. 2024. DOI: 10.48550/ARXIV.2407.02569.

[13] P. Chandarana, N. N. Hegade, K. Paul, F. Albarrán-Arriagada, E. Solano, A. del Campo, and X. Chen. "Digitized-counterdiabatic quantum approximate optimization algorithm." In: *Physical Review Research* 4.1 (Feb. 2022), p. 013141. ISSN: 2643-1564. DOI: 10.1103/physrevresearch.4.013141.

[14] A. A. Chaves, M. G. C. Resende, M. J. A. Schuetz, J. K. Brubaker, H. G. Katzgraber, E. F. de Arruda, and R. M. A. Silva. *A Random-Key Optimizer for Combinatorial Optimization*. 2024. DOI: 10.48550/ARXIV.2411.04293.

[15] J.-D. Cho, S. Raje, and M. Sarrafzadeh. "Fast approximation algorithms on maxcut, k-coloring, and k-color ordering for VLSI applications." In: *IEEE Transactions on Computers* 47.11 (1998), pp. 1253–1266. ISSN: 0018-9340. DOI: 10.1109/12.736440.

[16] P. W. Claeys, M. Pandey, D. Sels, and A. Polkovnikov. "Floquet-Engineering Counterdiabatic Protocols in Quantum Many-Body Systems." In: *Physical Review Letters* 123.9 (Aug. 2019), p. 090602. ISSN: 1079-7114. DOI: 10.1103/physrevlett.123.090602.

[17] K. De Jong and W. Spears. "On the virtues of parameterized uniform crossover." In: *Proceedings of the 4th Intern. Conf. on Genetic Algorithms, Morgan Kaufmann*. 1991.

[18] E. Farhi, D. Gamarnik, and S. Gutmann. *The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph: A Typical Case*. 2020. DOI: 10.48550/ARXIV.2004.09002.

[19] E. Farhi, J. Goldstone, and S. Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. DOI: 10.48550/ARXIV.1411.4028.

[20] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. *Quantum Computation by Adiabatic Evolution*. 2000. DOI: 10.48550/ARXIV.QUANT-PH/0001106.

[21] T. A. Feo and M. G. C. Resende. "Greedy Randomized Adaptive Search Procedures." In: *Journal of Global Optimization* 6.2 (Mar. 1995), pp. 109–133. ISSN: 1573-2916. DOI: 10.1007/bf01096763.

[22] J. R. Finžgar, A. Kerschbaumer, M. J. A. Schuetz, C. B. Mendl, and H. G. Katzgraber. "Quantum-Informed Recursive Optimization Algorithms." In: (2023). DOI: 10.48550/ARXIV.2308.13607.

[23] S. Foderà, G. Turati, R. Nembrini, M. F. Dacrema, and P. Cremonesi. *Reinforcement Learning for Variational Quantum Circuits Design*. 2024. DOI: 10.48550/ARXIV.2409.05475.

[24] M. Friedman. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance." In: *Journal of the American Statistical Association* 32.200 (Dec. 1937), pp. 675–701. ISSN: 1537-274X. DOI: 10.1080/01621459.1937.10503522.

[25] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. 27. print. A @series of books in the mathematical sciences. New York [u.a]: Freeman, 2009. 338 pp. ISBN: 9780716710448.

[26] J. F. Gonçalves and M. G. C. Resende. "Biased random-key genetic algorithms for combinatorial optimization." In: *Journal of Heuristics* 17.5 (Aug. 2010), pp. 487–525. ISSN: 1572-9397. DOI: 10.1007/s10732-010-9143-1.

[27] R. D. Guerrero. *Bee-yond the Plateau: Training QNNs with Swarm Algorithms*. 2024. DOI: 10.48550/ARXIV.2408.08836.

[28] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024.

[29] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas. "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz." In: *Algorithms* 12.2 (Feb. 2019), p. 34. ISSN: 1999-4893. DOI: 10.3390/a12020034.

[30] T. Hao, Z. He, R. Shaydulin, J. Larson, and M. Pistoia. *End-to-End Protocol for High-Quality QAOA Parameters with Few Shots*. 2024. DOI: 10.48550/ARXIV.2408.00557.

[31] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[32] IBM. *IBM ILOG CPLEX Optimization Studio 22.1.2 Documentation*. 2024.

[33] J. Kennedy and R. Eberhart. "Particle swarm optimization." In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. ICNN-95. IEEE, 1995, pp. 1942–1948. DOI: 10.1109/icnn.1995.488968.

[34] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980.

[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing." In: *Science* 220.4598 (May 1983), pp. 671–680. ISSN: 1095-9203. DOI: 10.1126/science.220.4598.671.

[36] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang. "The unconstrained binary quadratic programming problem: a survey." In: *Journal of Combinatorial Optimization* 28.1 (Apr. 2014), pp. 58–81. ISSN: 1573-2886. DOI: 10.1007/s10878-014-9734-0.

[37] M. Kölle, D. Seidl, M. Zorn, P. Altmann, J. Stein, and T. Gabor. *Optimizing Variational Quantum Circuits Using Metaheuristic Strategies in Reinforcement Learning*. 2024. DOI: 10.48550/ARXIV.2408.01187.

[38] Q. Langfitt, R. Tate, and S. Eidenbenz. *Phantom Edges in the Problem Hamiltonian: A Method for Increasing Performance and Graph Visibility for QAOA*. 2024. DOI: 10.48550/ARXIV.2411.05216.

[39] M. Larocca, P. Czarnik, K. Sharma, G. Muraleedharan, P. J. Coles, and M. Cerezo. "Diagnosing Barren Plateaus with Tools from Quantum Optimal Control." In: *Quantum* 6 (Sept. 2022), p. 824. ISSN: 2521-327X. DOI: 10.22331/q-2022-09-29-824.

[40] P. F. Leonhart, E. Spieler, R. Ligabue-Braun, and M. Dorn. "A biased random key genetic algorithm for the protein–ligand docking problem." In: *Soft Computing* 23.12 (Feb. 2018), pp. 4155–4176. ISSN: 1433-7479. DOI: 10.1007/s00500-018-3065-5.

[41] H. R. Lourenço, O. C. Martin, and T. Stützle. "Iterated Local Search." In: *Handbook of Metaheuristics*. Kluwer Academic Publishers, pp. 320–353. ISBN: 1402072635. DOI: 10.1007/0-306-48056-5_11.

[42] A. Lucas. "Ising formulations of many NP problems." In: (2013). DOI: 10.48550/ARXIV.1302.5843.

[43] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. "The theory of variational hybrid quantum-classical algorithms." In: *New Journal of Physics* 18.2 (Feb. 2016), p. 023023. ISSN: 1367-2630. DOI: 10.1088/1367-2630/18/2/023023.

[44] J. R. McClean, M. E. Kimchi-Schwartz, J. Carter, and W. A. de Jong. "Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states." In: *Physical Review A* 95.4 (Apr. 2017), p. 042308. ISSN: 2469-9934. DOI: 10.1103/physreva.95.042308.

[45] D. C. McKay, I. Hincks, E. J. Pritchett, M. Carroll, L. C. G. Govia, and S. T. Merkel. *Benchmarking Quantum Processor Performance at Scale*. 2023. DOI: 10.48550/ARXIV.2311.05933.

[46] J.-Y. P. Michel Gendreau, ed. *Handbook of Metaheuristics*. Springer US, 2010. ISBN: 9781441916655. DOI: 10.1007/978-1-4419-1665-5.

[47] N. Mladenović and P. Hansen. "Variable neighborhood search." In: *Computers & Operations Research* 24.11 (Nov. 1997), pp. 1097–1100. ISSN: 0305-0548. DOI: 10.1016/s0305-0548(97)00031-2.

[48] G. Nannicini. "Performance of hybrid quantum/classical variational heuristics for combinatorial optimization." In: (2018). DOI: 10.48550/ARXIV.1805.12037.

[49] J. A. Nelder and R. Mead. "A Simplex Method for Function Minimization." In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313. ISSN: 1460-2067. DOI: 10.1093/comjnl/7.4.308.

[50] P. B. Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

[51] A. C. M. d. Oliveira, A. A. Chaves, and L. A. N. Lorena. "Clustering search." In: *Pesquisa Operacional* 33.1 (Apr. 2013), pp. 105–121. ISSN: 0101-7438. DOI: 10.1590/s0101-74382013000100007.

[52] P. H. V. Penna, A. Subramanian, and L. S. Ochi. "An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem." In: *Journal of Heuristics* 19.2 (Sept. 2011), pp. 201–232. ISSN: 1572-9397. DOI: 10.1007/s10732-011-9186-y.

[53] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien. "A variational eigenvalue solver on a photonic quantum processor." In: *Nature Communications* 5.1 (July 2014). ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[54] M. J. D. Powell. "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation." In: *Advances in Optimization and Numerical Analysis*. Springer Netherlands, 1994, pp. 51–67. ISBN: 9789401583305. DOI: 10.1007/978-94-015-8330-5_4.

[55] J. Puerto, F. Ricca, M. Rodríguez-Madrena, and A. Scozzari. "A combinatorial optimization approach to scenario filtering in portfolio selection." In: *Computers & Operations Research* 142 (June 2022), p. 105701. ISSN: 0305-0548. DOI: 10.1016/j.cor.2022.105701.

[56] G. Reinelt. "TSPLIB—A Traveling Salesman Problem Library." In: *ORSA Journal on Computing* 3.4 (Nov. 1991), pp. 376–384. ISSN: 2326-3245. DOI: 10.1287/ijoc.3.4.376.

[57] S. Ropke and D. Pisinger. "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls." In: *European Journal of Operational Research* 171.3 (June 2006), pp. 750–775. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2004.09.004.

[58] M. J. A. Schuetz, J. K. Brubaker, H. Montagu, Y. van Dijk, J. Klepsch, P. Ross, A. Luckow, M. G. C. Resende, and H. G. Katzgraber. "Optimization of Robot Trajectory Planning with Nature-Inspired and Hybrid Quantum Algorithms." In: (2022). DOI: 10.48550/ARXIV.2206.03651.

[59] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran. "Evaluating analytic gradients on quantum hardware." In: (2018). DOI: 10.48550/ARXIV.1811.11184.

[60] M. Sciorilli, L. Borges, T. L. Patti, D. García-Martín, G. Camilo, A. Anandkumar, and L. Aolita. *Towards large-scale quantum optimization solvers with few qubits*. 2024. DOI: 10.48550/ARXIV.2401.09421.

[61] J. C. Spall. "A Stochastic Approximation Technique for Generating Maximum Likelihood Parameter Estimates." In: *1987 American Control Conference*. 1987, pp. 1161–1167. DOI: 10.23919/ACC.1987.4789489.

[62] K. J. Sung, J. Yao, M. P. Harrigan, N. C. Rubin, Z. Jiang, L. Lin, R. Babbush, and J. R. McClean. "Using models to improve optimizers for variational quantum algorithms." In: *Quantum Science and Technology* 5.4 (Sept. 2020), p. 044008. ISSN: 2058-9565. DOI: 10.1088/2058-9565/abb6d9.

[63] M. Suzuki. "Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems." In: *Communications in Mathematical Physics* 51.2 (June 1976), pp. 183–190. ISSN: 1432-0916. DOI: 10.1007/bf01609348.

[64] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[65] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles. "Noise-induced barren plateaus in variational quantum algorithms." In: *Nature Communications* 12.1 (Nov. 2021). ISSN: 2041-1723. DOI: 10.1038/s41467-021-27045-6.

[66] J. Wurtz and P. J. Love. "Counterdiabaticity and the quantum approximate optimization algorithm." In: *Quantum* 6 (Jan. 2022), p. 635. ISSN: 2521-327X. DOI: 10.22331/q-2022-01-27-635.

[67] J. Wurtz, S. Sack, and S.-T. Wang. *Solving non-native combinatorial optimization problems using hybrid quantum-classical algorithms*. 2024. DOI: 10.48550/ARXIV.2403.03153.

[68] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin. "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices." In: *Physical Review X* 10.2 (June 2020), p. 021067. ISSN: 2160-3308. DOI: 10.1103/physrevx.10.021067.

[69] L. Zhu, H. L. Tang, G. S. Barron, F. A. Calderon-Vargas, N. J. Mayhall, E. Barnes, and S. E. Economou. "Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer." In: *Physical Review Research* 4.3 (July 2022), p. 033029. ISSN: 2643-1564. DOI: 10.1103/physrevresearch.4.033029.