Forschungszentrum Jülich GmbH
John von Neumann Institute for Computing (NIC)

NIC Symposium 2022

29 – 30 September 2022 I Jülich, Germany

M. Müller, Ch. Peter, A. Trautmann (Editors)

Proceedings

Publication Series of the John von Neumann Institute for Computing (NIC)
NIC Series Volume 51

ISBN 978-3-95806-646-5

Bibliografische Information der Deutschen Nationalbibliothek.
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte Bibliografische Daten
sind im Internet über http://dnb.d-nb.de abrufbar.

Herausgeber und Vertrieb:	Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 52425 Jülich Tel.: +49 2461 61-5368 Fax: +49 2461 61-6103 zb-publikation@fz-juelich.de www.fz-juelich.de/zb
Umschlaggestaltung:	Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH
Druck:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2022

We thank J. Schumacher and Ph. Vieweg (Technische Universität Ilmenau) for the image used in the cover design.

Publication Series of the John von Neumann Institute for Computing (NIC)
NIC Series Volume 51

ISBN 978-3-95806-646-5

Vollständig frei verfügbar über das Publikationsportal des Forschungszentrums Jülich (JuSER)
unter www.fz-juelich.de/zb/openaccess.



This is an Open Access publication distributed under the terms of the Creative Commons Attribution License 4.0, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Hybrid Quantum Classical Simulations

Dennis Willsch[1], **Manpreet Jattana**[1,2], **Madita Willsch**[1,3], **Sebastian Schulz**[1,2], **Fengping Jin**[1], **Hans De Raedt**[1,4], **and Kristel Michielsen**[1,2,3]

[1] Institute for Advanced Simulation, Jülich Supercomputing Centre, Forschungszentrum Jülich, 52425 Jülich, Germany
E-mail: {d.willsch, m.jattana, m.willsch, se.schulz, f.jin, k.michielsen}@fz-juelich.de

[2] RWTH Aachen University, 52056 Aachen, Germany

[3] AIDAS, 52425 Jülich, Germany

[4] Zernike Institute for Advanced Materials, University of Groningen, Nijenborgh 4, NL-9747 AG Groningen, The Netherlands
E-mail: deraedthans@gmail.com

We report on two major hybrid applications of quantum computing, namely, the quantum approximate optimisation algorithm (QAOA) and the variational quantum eigensolver (VQE). Both are hybrid quantum classical algorithms as they require incremental communication between a classical central processing unit and a quantum processing unit to solve a problem. We find that the QAOA scales much better to larger problems than random guessing, but requires significant computational resources. In contrast, a coarsely discretised version of quantum annealing called approximate quantum annealing (AQA) can reach the same promising scaling behaviour using much less computational resources. For the VQE, we find reasonable results in approximating the ground state energy of the Heisenberg model when suitable choices of initial states and parameters are used. Our design and implementation of a general quasi-dynamical evolution further improves these results.

1 Introduction

Quantum computing[1] is an emerging computer technology that uses quantum effects in the design of its computational model. There are two major paradigms in quantum computing, namely the gate-based quantum computer and the quantum annealer.

A gate-based quantum computer is inspired by the circuit model for classical, digital computers. This means that every program is defined in terms of a sequence of fundamental operations, the so-called *quantum gates*. Each quantum gate operates on the fundamental units of computation, the so-called *quantum bits* or *qubits*. The gate-based quantum computer executes these quantum gates step-by-step and thus updates its internal quantum state. At the end of the quantum gate sequence, the quantum state is "measured", which means that the quantum computer outputs, with a certain probability, one out of several classical bitstrings. Typically, the number of bits in this bitstring is given by the number of qubits in the quantum computer. At the time of writing, gate-based quantum computers with approximately one hundred qubits have been manufactured[2].

A quantum annealer, on the other hand, tries to harness the natural evolution of a quantum system, steered by external magnetic fields, to solve an optimisation problem. Internally, it also operates on a set of qubits, which are measured at the end of a quantum annealing run. It thus also produces a bitstring as output. At the time of writing, quantum annealers with more than 5000 qubits have been manufactured[3].

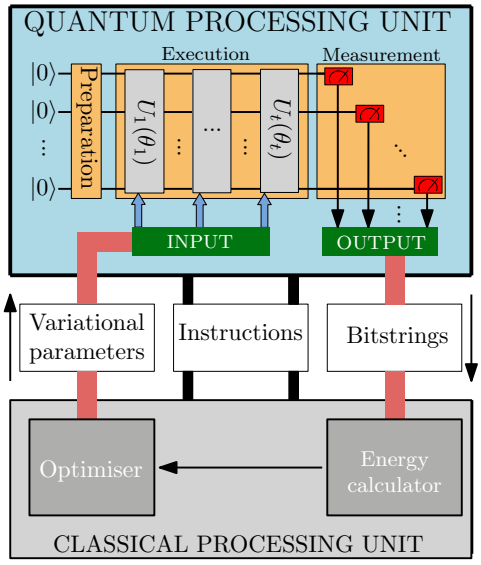

Figure 1. **Schematic diagram of hybrid quantum classical variational algorithms.** The tasks are divided between a QPU and a CPU. The CPU suggests certain parameters of a parametrised quantum circuit. This quantum circuit, along with other instructions, is then sent to and executed by the QPU. Following this, the measurement results from the QPU are sent back to the CPU, which in turn evaluates the energy to be optimised. From this result, the CPU obtains new parameters for the quantum circuit. This process continues until some convergence criterion is met.

Both types of quantum computers share the property that the same program can produce different bitstrings obtained after a run. The computational models are thus inherently probabilistic. This means that quantum computers can also be used as *samplers*. A program is often executed multiple times to obtain a representative distribution of bitstrings.

In this article, we consider simulations of hybrid quantum classical variational algorithms. Such algorithms obtain their name from the combined usage of gate-based quantum and classical computers in a single application. The working of such algorithms is visualised in Fig. 1. The optimiser in the classical central processing unit (CPU) sends the parameters placed in a circuit to be executed by the quantum processing unit (QPU). The QPU prepares a problem specific initial state, executes the circuit, and sends the results to the CPU after measurement. The outcomes of the measurements, called bitstrings, are used to calculate the energy which is given to the optimiser. The optimiser then decides what new parameters will lower the energy, and the cycle continues until convergence. The CPU also controls the QPU through other instructions, e.g. the number of measurements, microwave pulses, or the time between measurements.

This article is structured as follows. Sec. 2 contains an overview of the Jülich Universal Quantum Computer Simulator, which is used for the simulation of the following two hybrid quantum classical applications. In Sec. 3, we introduce the QAOA, discuss its relation to AQA, and compare both algorithms when applied to the tail assignment problem. In Sec. 4, we discuss the VQE and its application to the Heisenberg model. Sec. 5 contains our conclusions.

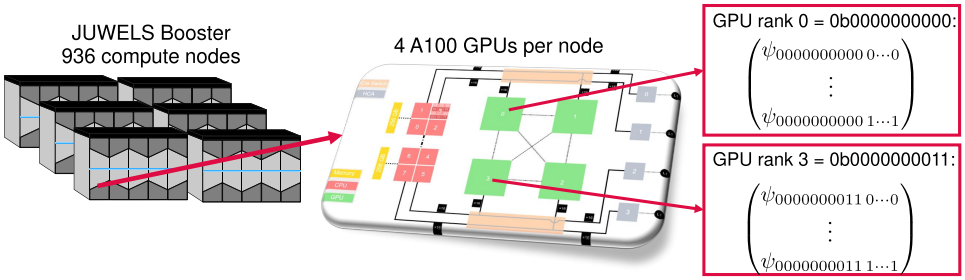

Figure 2. **Distribution of the coefficients of the state vector $|\psi\rangle$ across the GPUs and compute nodes of JUWELS Booster[7].** Each GPU is handled by a single MPI process. For each GPU, the global qubit indices of the coefficients represent its MPI rank. For the GPUs belonging to MPI rank 0 and 3 this is indicated on the right (the 10 leftmost indices are the global qubit indices). The complex coefficients of the local qubits are stored locally on the GPUs.

2 Jülich Universal Quantum Computer Simulator

In this section, we briefly describe the Jülich Universal Quantum Computer Simulator (JUQCS), and outline how a program for a gate-based quantum computer can be simulated using supercomputers. More details and in-depth descriptions of the implementation are given in Refs. 4–6.

The basic unit of computation for quantum computers is the qubit. Mathematically, a qubit is described by a unit vector of two complex numbers $|\psi\rangle = (\psi_0, \psi_1)$ fulfilling $\langle\psi|\psi\rangle = |\psi_0|^2 + |\psi_1|^2 = 1$. Usually, the basis states (i.e., orthonormal basis vectors) are denoted by $|0\rangle$ and $|1\rangle$. An N-qubit system is defined by 2^N complex numbers

$$|\psi\rangle = \psi_{0\ldots00}\,|0\ldots00\rangle + \psi_{0\ldots01}\,|0\ldots01\rangle + \ldots + \psi_{1\ldots11}\,|1\ldots11\rangle\,, \tag{1}$$

where $|0\ldots00\rangle, \ldots, |1\ldots11\rangle$ denote the computational basis states[1] and the complex coefficients $\psi_{0\ldots00}, \ldots, \psi_{1\ldots11}$ again fulfil $\langle\psi|\psi\rangle = 1$. The 2^N complex coefficients in the state $|\psi\rangle$ can be written as a rank-N tensor $\psi_{q_{N-1}\cdots q_1 q_0}$ where $q_j \in \{0, 1\}$ denote the indices.

Since the number of coefficients grows exponentially in the number of qubits N, the memory requirement grows exponentially too, which makes large-scale simulations of universal quantum computers only possible on supercomputers with enough (distributed) random access memory. To simulate for instance a quantum computer with $N = 42$ qubits, the tensor $\psi_{q_{N-1}\cdots q_1 q_0}$ requires $16 \times 2^{42}\,\mathrm{B} = 64\,\mathrm{TiB}$ of memory (using double precision floating-point numbers) while the simulation of $N = 21$ qubits only requires $16 \times 2^{21}\,\mathrm{B} = 32\,\mathrm{MiB}$ of memory.

The GPU version of JUQCS (JUQCS–G) distributes the complex coefficients over the memory of the participating GPUs. A sketch is shown in Fig. 2. Each GPU stores a power of two coefficients, say 2^M coefficients, of the state vector $|\psi\rangle$ in its local memory. Qubits j whose amplitudes $\psi_{q_{N-1}\ldots q_{j+1}0q_{j-1}\ldots q_0}$ and $\psi_{q_{N-1}\ldots q_{j+1}1q_{j-1}\ldots q_0}$ are stored on the same local memory are thus called *local* qubits. Qubits whose amplitudes are distributed over different GPUs are called *global* qubits. The total number of GPUs needed is $N_{\mathrm{GPU}} = 2^{N-M}$.

Since the state vector is distributed over several GPUs, potentially also over different compute nodes, it is necessary to transfer data from one GPU to another over the network, for instance if (part of) the data is needed for a computation on another GPU or node. We use the *Message Passing Interface* (MPI) to transfer the data between different GPUs. Each MPI process controls a GPU, and its rank $r \in \{0, \ldots, N_{\mathrm{GPU}} - 1\}$ is given by the global qubit indices in binary notation.

The application of a quantum gate is performed as the matrix-vector update $U|\psi\rangle$, where the matrix U is unitary and acts only on a few-qubit subspace. For instance, the matrix corresponding to a single-qubit gate is a 2×2 unitary matrix $U_1 = (u^{(1)}_{qq'})$. It acts only on a single-qubit subspace and thus transforms the coefficients of $|\psi\rangle$ in terms of 2-component updates: Acting on qubit j, U_1 transforms the state vector $|\psi\rangle$ according to

$$\psi_{q_{N-1}\cdots q_{j+1} q q_{j-1}\cdots q_0} \leftarrow \sum_{q'=0}^{1} u^{(1)}_{qq'} \psi_{q_{N-1}\cdots q_{j+1} q' q_{j-1}\cdots q_0}\,, \tag{2}$$

for $q = 0, 1$. Similarly, a two-qubit gate is a 4×4 unitary matrix that acts on a two-qubit subspace of $|\psi\rangle$, and so on. A suitable set of one- and two-qubit gates (called a universal gate set) is sufficient for the implementation of a universal quantum computer (simulator)[8,9]. The full gate set of JUQCS is documented in Ref. 5.

The implementation of JUQCS exploits the special structure of single-, two- and three-qubit operations. Large, dense matrices are never stored nor operated on. It is sufficient to loop over pairs of coefficients of the state vector and to perform the 2-component updates given in Eq. 2 for the same 2×2 matrix U_1 (how U_1 looks like depends on the particular gate). Depending on the index of the qubit that the gate operates on, the pairs of coefficients are grouped differently. Accordingly, for two-qubit gates, the loop goes over quadruples of coefficients of $|\psi\rangle$ and the 4-component updates are performed with a 4×4 matrix (whose entries are determined by the particular gate).

A quantum gate acting on a global qubit requires MPI communication, because coefficients of $|\psi\rangle$ that need to be combined in the update are stored on different GPUs. Thus, to perform a single-qubit gate on a global qubit, the transfer of $2^N/2$ coefficients (i.e., half of the state vector) between pairs of GPUs is required. The communication overhead is minimised by *relabelling* global and local qubits after the transfer happened once. Thus, the coefficients do not need to be transferred back again after the transformation. Each MPI rank has to keep track of the current labelling of global and local qubits. More details on this MPI communication scheme are given in Ref. 4.

3 Quantum Approximate Optimisation Algorithm

The QAOA[10] is an algorithm for gate-based quantum computers to solve combinatorial optimisation problems. Such problems can be mapped to binary optimisation problems, which are defined in terms of an N-bit binary string $\mathbf{x} = x_1 \ldots x_N$, with the goal of finding a bitstring $\mathbf{x}$ that minimises a classical objective function $C(\mathbf{x}) : \{0,1\}^N \to \mathbb{R}$.

For quantum computers, a natural way of encoding a binary optimisation problem is to express its solution as the ground state (i.e., the state corresponding to the lowest energy)

of the so-called Ising Hamiltonian

$$H_{\mathrm{C}} = \sum_{i=0}^{N-1} h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z. \tag{3}$$

Here, $\{h_i\}$ and $\{J_{ij}\}$ denote real numbers that represent the optimisation problem instance, and $\sigma_i^z = I \otimes I \cdots I \otimes \mathrm{diag}(1,-1) \otimes I \otimes \cdots \otimes I$ (with I being the 2×2 identity matrix) is the Pauli z matrix acting on the ith qubit subsystem.

The idea of the QAOA is to solve the optimisation problem by bringing the quantum computer into a variational state

$$|\beta, \gamma\rangle = \left(\prod_{k=p}^{1} \exp\left[-i\beta_k H_{\mathrm{D}}\right] \exp\left[-i\gamma_k H_{\mathrm{C}}\right] \right) |+\rangle^{\otimes N}, \tag{4}$$

where $\gamma = (\gamma_1, ..., \gamma_p)$ and $\beta = (\beta_1, ..., \beta_p)$ are the $2p$ variational parameters to be optimised, $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ is the uniform superposition of the qubit state $|0\rangle$ and $|1\rangle$, H_C is the cost Hamiltonian encoding the optimisation problem (see Eq. 3), and $H_{\mathrm{D}} = -\sum_{i=0}^{N-1} \sigma_i^x$ is a so-called driving Hamiltonian (i.e., a sum of Pauli x matrices). By finding optimal angles β and γ that minimise the energy expectation value $E_p(\beta, \gamma) = \langle \beta, \gamma | H_{\mathrm{C}} | \beta, \gamma \rangle$, one hopes to find the ground state of H_{C}. As $|\beta, \gamma\rangle$ is a linear combination of 2^N potential states, the idea is that optimising the $2p$ variational parameters may be much simpler than solving the original optimisation problem. However, note that in general, determining a set of optimal β and γ may require numerous queries to a quantum computer. A quantum circuit to bring the quantum computer into the variational state $|\beta, \gamma\rangle$ given by Eq. 4 is shown in Fig. 3.

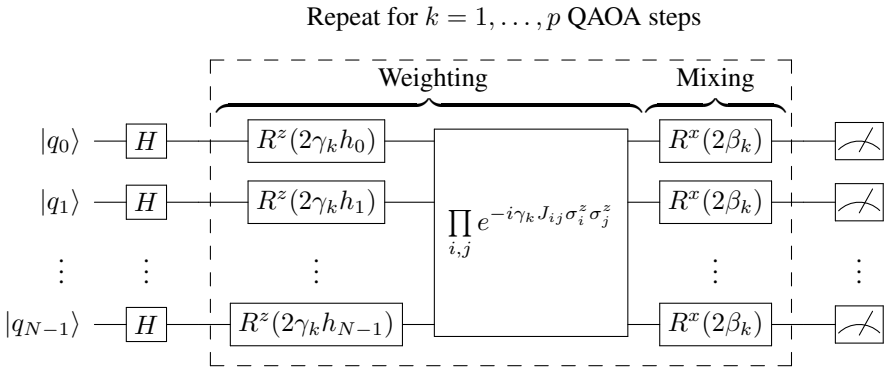

Figure 3. **General QAOA circuit.** Initially, we bring the qubits into a uniform superposition over all states using the Hadamard gate H. Then, we apply $k = 1, \ldots, p$ QAOA steps with parameters $\beta_1, \ldots, \beta_p$ and $\gamma_1, \ldots, \gamma_p$ (see Eq. 4). Each QAOA step k consists of a "weighting step" using single-qubit z rotation gates $R^z(\varphi) = \exp(-i\varphi\sigma^z/2)$ and the set of two-qubit gates indicated in the circuit, followed by a "mixing step" using single-qubit x rotation gates $R^x(\varphi) = \exp(-i\varphi\sigma^x/2)$. The result after the measurement gates is used to update the variational parameters until the energy is sufficiently low (cf. Fig. 1).

3.1 Emulation of the QAOA

To emulate the QAOA, JUQCS produces the variational state $|\beta, \gamma\rangle$ (see Eq. 4) by simulating the quantum circuit in Fig. 3 for given variational parameters β and γ. For the optimisation phase of the QAOA, the energy expectation value in the variational state $E_p(\beta, \gamma) = \langle\beta, \gamma| H_C |\beta, \gamma\rangle$ is computed by JUQCS and passed to the optimiser (cf. Fig. 1; we use optimisers from the `scipy` library[11]). The optimiser then returns new values for the variational parameters, which are in turn used for the next simulation of the QAOA circuit, and so on. In case the optimisation does not converge or if it converges too slowly, an additional stopping criterion of a maximum of 200 calls to JUQCS is used.

Note that in practice, the only quantity that can be estimated using a real quantum computer is the energy expectation value $E_p(\beta, \gamma)$, and not the success probability P_{success} (i.e., the probability of the ground state in the variational state $|\beta, \gamma\rangle$). Although we could in principle also optimise for the success probability, we consider the realistic situation and optimise for the energy only. However, the success probability is also evaluated by JUQCS to assess the quality of the solution (cf. Fig. 4 below).

3.2 Approximate Quantum Annealing

In this section, we first review how to simulate the time evolution of a quantum annealing process, and then we explain under which conditions we refer to "approximate" quantum annealing (AQA). Additionally, we point out the structural relation to the QAOA introduced above. A more detailed discussion of AQA and its comparison to the QAOA is given in Ref. 6.

In order to simulate a quantum annealing process, the time-dependent Schrödinger equation (TDSE) with $\hbar = 1$, $i\partial_t |\psi(t)\rangle = H(t) |\psi(t)\rangle$, for the quantum annealing Hamiltonian $H(t) = A(t)H_{\text{D}} + B(t)H_{\text{C}}$ has to be solved at times $0 \leq t \leq t_{\text{anneal}}$, where t_{anneal} is the annealing time. Here, H_D (H_C) is the driving (cost) Hamiltonian used in Eq. 4, and the so-called annealing functions A and B satisfy $A(0) \gg B(0)$ and $A(t_{\text{anneal}}) \ll B(t_{\text{anneal}})$. Therefore, we also call $H(0) \approx H_D$ and $H(t_{\text{anneal}}) \approx H_C$ the initial and final Hamiltonian, respectively.

The TDSE can be solved by time-stepping using the second-order Suzuki-Trotter product formula[12,13]

$$|\Psi((l+1)\tau)\rangle = \left\{ \exp\left[-\frac{i\tau}{2} A(l\tau) H_D\right] \times \exp[-i\tau B(l\tau) H_C] \right. \\ \left. \times \exp\left[-\frac{i\tau}{2} A(l\tau) H_D\right] \right\} |\Psi(l\tau)\rangle \,, \tag{5}$$

where τ denotes the time step and $l = 0, \ldots, n$ (such that $t_{\text{anneal}} = (n+1)\tau$). The action of the three matrix exponentials in Eq. 5 on any state vector can be computed analytically. The initial state is taken to be the ground state of H_{D}, namely $|\Psi(0)\rangle = |+\rangle^{\otimes N}$. Note that the structure of Eq. 5 and of the QAOA (cf. Eq. 4) are essentially the same.

The idea of AQA is now to use a time step τ that is too large and a number of time steps n that is too small to yield an accurate discretisation of the time evolution. Thus, we do not expect that the sequence of states $|\Psi((l+1)\tau)\rangle$ resembles the solution of the time evolution of a quantum annealing process. However, we hope that we can still generate a final state which has a relatively large overlap with the ground state of the final problem

Hamiltonian H_C by using only a small number of matrix-vector operations. This heuristic method is motivated by empirical findings[14–18] that the values for β and γ obtained from the optimisation were often found to approximately follow curves that are suitable as coarsely discretised quantum annealing functions A and B.

The amounts of single-qubit and two-qubit gates required to implement either one time step of AQA, or one layer of the QAOA, are the same. Thus, for fixed n, (i.e., $n+1$ AQA time steps) the computational effort is equivalent to a single QAOA circuit evaluation (i.e., no optimisation) where $p = n + 1$.

3.3 Tail Assignment Problem

The tail assignment problem[19] is a formulation of aircraft assignment problems that respects typical operational constraints such as connection times, preassigned activities, maintenance and airport curfews. We consider the constraint-only version of the tail assignment problem used in Ref. 20 (other models for the tail assignment problem can be found in Refs. 18, 21, 22). In this form, the problem can be formulated as an exact cover problem, which is an NP-complete problem[23] from the class of set covering and partitioning problems. The corresponding classical objective function to be minimised reads

$$C(\mathbf{x}) = \sum_{f=1}^{F} \left(\sum_{i=1}^{N} \mathbf{A}_{if} x_i - 1 \right)^2 . \tag{6}$$

Here, $\mathbf{A} \in \{0,1\}^{N \times F}$ is a Boolean matrix that defines the exact cover problem instance. In the context of aircraft assignment, each of the F columns of $\mathbf{A}$ represents one flight, and each of the N rows represents a given route. Thus, $\mathbf{A}_{if} = 1$ if and only if flight f is contained in route i. The solution $\mathbf{x} = x_1 \ldots x_N$ therefore selects a set of aircraft routes from $\mathbf{A}$ such that each flight is covered exactly once (see Ref. 20, which also contains the explicit expression for H_C in Eq. 3 as a function of $\mathbf{A}$). We consider the same $N = 30, \ldots, 40$ qubit problems with $F = 472$ flights that have been used in Ref. 6.

3.4 Results

In this section, we summarise some representative QAOA and AQA results. In Fig. 4, we plot the success probability P_{success} as a function of the problem size N for two versions of AQA and the QAOA. The AQA results are for $n = 5$ steps with time step $\tau = 0.1$ (green squares) and $\tau = 0.8$ (green crosses). The QAOA results are for standard QAOA (i.e., the variational parameters are optimised individually for each problem instance, initialised from AQA with $n = 6$ and $\tau = 0.4$; blue circles), and for pre-optimised QAOA (i.e., the variational parameters are optimised for a 30-qubit problem, initialised from AQA with $n = 5$ and $\tau = 0.2$, and these same values are reused for all larger problem instances; blue triangles). Further results and additional details can be found in Ref. 6.

Both QAOA and AQA obtain better success probabilities than can be found by random guessing (red dash-dot-dotted line), but for the QAOA they are overall higher than for AQA. This is reasonable since the variational QAOA parameters have been initialised with AQA results. More important, however, is the scaling behaviour as a function of the problem size N. For AQA with $\tau = 0.1$ (dash-dotted line), the scaling is equal to the scaling of random guessing ($1/2^N$). The scaling of the QAOA with pre-optimised variational

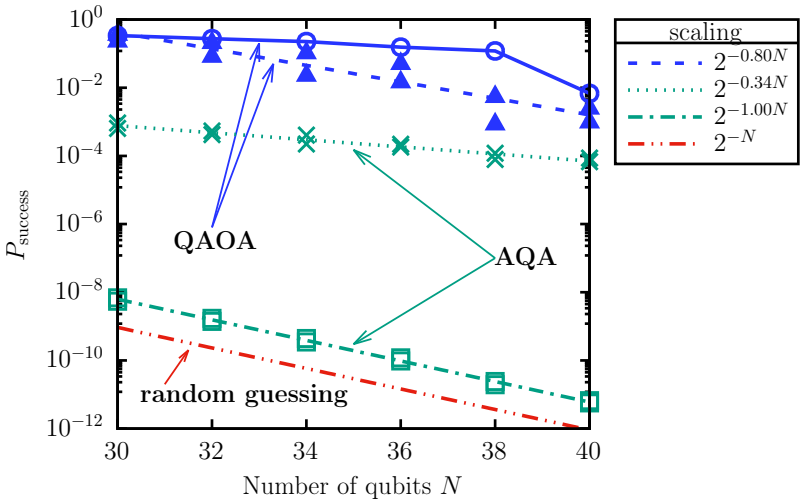

Figure 4. **Success probabilities P_{success} obtained from AQA and the QAOA, in comparison to random guessing, for different problem sizes N.** The success probabilities obtained with the QAOA are shown for a case in which the variational parameters are separately optimised for each problem size N (blue circles, $p = 7$), and for a case in which fixed variational parameters are taken from a previous optimisation of the $N = 30$ case (blue triangles, $p = 6$). The success probabilities obtained with AQA are shown for $\tau = 0.8$ (green crosses) and $\tau = 0.1$ (green squares). The dashed, dotted, and dash-dotted lines represent fits to the average success probabilities (in these cases, two instances per problem size have been simulated). The red dash-dot-dotted line represents the probability $1/2^N$ to find the solution by random guessing. The solid line is a guide to the eye.

parameters (dashed line) is only marginally better. However, both AQA with $\tau = 0.8$ (dotted line) and standard QAOA (solid line) show considerably good scaling behaviour (albeit still exponential).

Merely the particular QAOA data point for the largest problem size (the blue circle at $N = 40$) falls out of this scaling behaviour. Here, the maximum number of 200 JUQCS-calls (and thus the limit of the computational budget allocated for this job) was reached. This represents the fact that the QAOA optimisation procedure requires a significant amount of computational work on many nodes of the supercomputer.

This is a noteworthy advantage of AQA, as it can extend the good scaling also to $N = 40$ without requiring any involved optimisation procedure once a good value for τ is found. Interestingly, the regime of "good" values appears to be for large time steps τ, where AQA is actually not a genuine discretisation of quantum annealing anymore.

4 Variational Quantum Eigensolver

An interesting and common problem is to find the ground state energy of a Hamiltonian H. As this is a hard problem to solve on classical computers, quantum computers are considered as an alternative route to solve the problem. When the ground state $|\psi_0\rangle$ can be

prepared on a quantum computer, the expectation value

$$\langle H \rangle_0 = \frac{\langle \psi_0 | H | \psi_0 \rangle}{\langle \psi_0 | \psi_0 \rangle} \tag{7}$$

gives the ground state energy. The task is to find this ground state energy by initialising the quantum computer in an arbitrary initial trial state $|\psi_t\rangle$. The variational theorem states that $\langle H \rangle_t$ for any $|\psi_t\rangle \neq |\psi_0\rangle$ is an upper bound to the ground state energy. This means that

$$\langle H \rangle_t \geq E_0, \tag{8}$$

where the equality sign holds if and only if $|\psi_t\rangle = |\psi_0\rangle$. In Eq. 8, E_0 is the lowest eigenvalue or the ground state energy of the physical system described by H.

The VQE is an implementation of the variational principle on a quantum computer. Its working is illustrated in Fig. 1. The VQE derives its name from the fact that it was used to solve for the lowest eigenvalue of a Hamiltonian[24].

4.1 Heisenberg Model

The Hamiltonian for the one-dimensional Heisenberg model is given by

$$H = \sum_{i>j}^{N} \Big(J_{ij}^{xx} \sigma_i^x \cdot \sigma_j^x + J_{ij}^{yy} \sigma_i^y \cdot \sigma_j^y + J_{ij}^{zz} \sigma_i^z \cdot \sigma_j^z \Big), \tag{9}$$

where N denotes the number of spins, σ^x, σ^y, and σ^z are the Pauli matrices, and J_{ij} is the coupling interaction between the ith and j^{th} spins. We use units such that $\hbar = 1$ and the J's are dimensionless. If $J_{ij} = 1$ for all $i, j = 1, \ldots, N$, we call H the *isotropic antiferromagnetic* model Hamiltonian. We consider bipartite spin lattices in our simulations.

4.1.1 The Ansatz

The variational principle allows any ansatz to be used to solve the ground state energy problem. However, not every ansatz can express the ground state of H. An ansatz[25] that can find a reasonable approximation to the ground state energy is given by

$$U(\boldsymbol{\theta}) = \Big[\prod_{l=N-1}^{1} \prod_{k=N}^{l+1} U_{lk}(\theta_{lk}) \Big] \Big[\prod_{l=N-1}^{1} \prod_{k=N}^{l+1} U_{kl}(\theta_{kl}) \Big], \tag{10}$$

where

$$U_{pq}(\theta_{pq}) = \begin{cases} e^{-i\theta_{pq}\sigma_p^y \sigma_q^x} & \text{if } p = N \text{ or } q = N, \\ e^{-i\theta_{pq}\sigma_p^y \sigma_q^x \sigma_N^z} & \text{otherwise.} \end{cases} \tag{11}$$

If an ansatz can express the ground state, it is still challenging to optimise the parameters such that $|\psi_t(\boldsymbol{\Theta})\rangle = |\psi_0\rangle$, where $\boldsymbol{\Theta}$ are the optimised parameters. In addition to the ansatz, it is important to specify the initial state (say $|\Psi_0\rangle$) of the quantum computer. The ansatz then acts on the initial state, i.e. $\boldsymbol{U}(\boldsymbol{\theta}) |\Psi_0\rangle$, with a set of initial parameters to give the initial energy. The task of the VQE is to lower this energy until it equals the ground state energy. A better choice of initial energy can improve the performance of the VQE. To obtain a better (than random) initial energy for the Heisenberg model, it helps to set the

initial state to the Néel state and the initial parameters to zero. On a lattice with an even number of spins, the Néel state is prepared by setting adjacent spins anti-parallel. On a quantum computer, and for a one dimensional model, this is achieved by setting the qubits with the odd (even) indices to $|0\rangle$ ($|1\rangle$), or *vice versa*.

4.1.2 Results

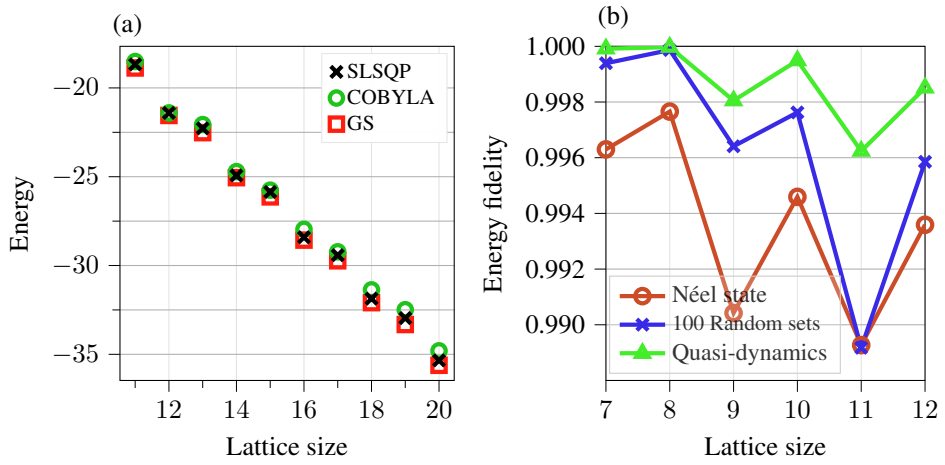

Figure 5. (a) Optimised variational energies obtained using gradient based (SLSQP) and gradient free (COBYLA) algorithms in comparison to the ground state energies for different lattice sizes. (b) Comparison between the energy fidelities obtained using the Néel initial state, best of 100 random sets of parameters, and quasi-dynamics.

We find the ground state energy of isotropic Heisenberg rings of length 11 to 20 using two different optimisers. We used the gradient based (free) optimiser SLSQP (COBYLA)[26,11,27]. The results are compared against the ground state energy found by the Lanczos algorithm and are shown in Fig. 5(a). We observe, as expected, that using different optimisers leads to different results. SLSQP performs better in all cases but the difference in obtained energies appears to grow with increasing lattice sizes.

4.2 VQE and Quasi-Dynamics

When using the VQE, the variational quantum state as a function of m parameters is written as

$$|\psi\rangle = U(\boldsymbol{\theta})\,|\Psi_0\rangle = U_m(\theta_m)\ldots U_1(\theta_1)\,|\Psi_0\rangle\,. \tag{12}$$

If $\mathbb{U}(\boldsymbol{\Theta})$ ($U(\boldsymbol{\theta})$) represents the unitary operators corresponding to the optimised (unoptimised) numeric values of the parameters, then the VQE can be said to perform the task $U(\boldsymbol{\theta}) \to \mathbb{U}(\boldsymbol{\Theta})$. After the optimiser signals convergence, the state obtained is

$$|\Psi_1\rangle = \mathbb{U}(\boldsymbol{\Theta})\,|\Psi_0\rangle\,. \tag{13}$$

The idea is to use this state as an initial state for another round of VQE optimisation. To do so, substitute Eq. 13 in Eq. 12 repeatedly (p times), such that the final state is

$$|\Psi_p\rangle = \mathbb{U}_p(\boldsymbol{\Theta}_p) \ldots \mathbb{U}_1(\boldsymbol{\Theta}_1) |\Psi_0\rangle . \tag{14}$$

The procedure represented by Eq. 14 is termed *quasi-dynamical evolution*[28]. A suitable choice of initial parameters for each $U(\boldsymbol{\theta})$ is $\boldsymbol{\theta} = [0, \ldots, 0]$. This step is necessary in order to preserve the progress made until then. By avoiding random initialisation we also avoid potential barren plateaus[29]. Similar to the QAOA, the performance of the quasi-dynamics is expected to improve as p is increased.

4.2.1 Results

We perform the quasi-dynamical evolution on isotropic rings of lengths 7 to 12. We set the threshold for the quasi-dynamics such that the process stops when the energy at step p can no longer be lowered below 10^{-4} compared to the $(p-1)^{\text{th}}$ value. The results are compared against the final energy obtained when (1) the Néel initial state and (2) one hundred random sets of initial parameters were used. We define the energy fidelity as the ratio of the variational energy obtained upon convergence of the optimiser to the ground state energy. For (2), we plot only the highest energy fidelity obtained from all cases. The results are shown in Fig. 5(b). We observe that the quasi-dynamical evolution can find a better energy fidelity.

In our results we used the same $\boldsymbol{U}(\boldsymbol{\theta})$ (from Eq. 10) for each of the p iterations in Eq. 14. While it improves the results for the cases we tested, such an elementary choice may not always converge to the ground state energy. Eq. 14 allows the use of any arbitrary $\boldsymbol{U}(\boldsymbol{\theta})$; however, it remains an open question which choices of $\boldsymbol{U}(\boldsymbol{\theta})$ will find the ground state energy for problems in general.

We observed that the number of random restart cases that found an energy better than the energy found when starting from the Néel state dropped as the lattice size is increased (data not shown). Since the ansatz has more parameters for larger lattices, one solution is to increase the number of restarts. However, this approach can be very resource expensive. Thus, random initialisation is not a suitable strategy. It is crucial to have a good combination of ansatz and initial parameters to have a reasonable chance of finding the ground state energy.

5 Conclusions

In this article, we have discussed two popular hybrid quantum classical algorithms, namely the QAOA and the VQE. Both have been simulated on JUWELS Booster using JUQCS–G.

For the QAOA, we found that the best scaling of the success probability can be obtained when the full variational optimisation procedure is carried out. However, due to tremendous computational resources required for simulations with $N = 40$ qubits, it was not possible to reach convergence for the largest problem size. Therefore, we considered AQA as an alternative algorithm with only a single parameter τ to solve this problem. Although slightly worse in terms of actual success probabilities, AQA could extend the same promising scaling behaviour also to the largest problems. Interestingly, the resulting

optimal τ was so large that it put AQA into a regime where it does not accurately describe a discretised quantum annealing process anymore.

For the VQE, we found that a suitable choice of an ansatz coupled with a good choice of initial parameters is critical for finding the ground state energies. For the Heisenberg model, one such choice is the Néel state with all variational parameters initialised to zero. However, which choices are helpful in general is an open question. We introduced and tested quasi-dynamical evolution, a technique which builds upon the VQE and improves its performance. An open question for the quasi-dynamics is the choice of $\boldsymbol{U}$ in each substitution which guarantees finding the ground state energy.

Acknowledgements

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (`www.gauss-centre.eu`) for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC). We would like to thank M. Svensson for providing the exact cover problem instances. D.W. and M.W. acknowledge support from the project Jülich UNified Infrastructure for Quantum computing (JUNIQ) that has received funding from the German Federal Ministry of Education and Research (BMBF) and the Ministry of Culture and Science of the State of North Rhine-Westphalia. M. J. acknowledges support from the project OpenSuperQ (820363) of the EU Quantum Flagship.

References

1. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, New York, 2010.
2. IBM Q team, "127-qubit backend: IBM Q 127 Washington backend specification V1.1.0", 2022.
3. C. McGeoch and P. Farré, *The D-Wave Advantage System: An Overview*, Tech. Rep., D-Wave Systems Inc, Burnaby, BC, Canada, 2020, D-Wave Technical Report Series 14-1049A-A.
4. K. De Raedt, K. Michielsen, H. De Raedt, B. Trieu, G. Arnold, M. Richter, T. Lippert, H. Watanabe, and N. Ito, *Massively parallel quantum computer simulator*, Comput. Phys. Commun., **176**, no. 2, 121, 2007.
5. H. De Raedt, F. Jin, D. Willsch, M. Willsch, N. Yoshioka, N. Ito, S. Yuan, and K. Michielsen, *Massively parallel quantum computer simulator, eleven years later*, Comput. Phys. Commun., **237**, 47–61, 2019.
6. D. Willsch, M. Willsch, F. Jin, K. Michielsen, and H. De Raedt, *GPU-accelerated simulations of quantum annealing and the quantum approximate optimization algorithm*, Comput. Phys. Commun., **278**, 108411, 2022.
7. S. Kesselheim, A. Herten, K. Krajsek, J. Ebert, J. Jitsev, M. Cherti, M. Langguth, B. Gong, S. Stadtler, A. Mozaffari, G. Cavallaro, R. Sedona, A. Schug, A. Strube, R. Kamath, M. G. Schultz, M. Riedel, and T. Lippert, "JUWELS Booster – A Supercomputer for Large-Scale AI Research", 2021.
8. D. Deutsch, A. Barenco, and A. Ekert, *Universality in Quantum Computation*, Proc. R. Soc. Lond. A, **449**, no. 1937, 669, 1995.

9. D. P. DiVincenzo, *Two-bit gates are universal for quantum computation*, Phys. Rev. A, **51**, 1015–1022, Feb 1995.
10. E. Farhi, J. Goldstone, and S. Gutmann, *A Quantum Approximate Optimization Algorithm*, 2014, arXiv:1411.4028.
11. E. Jones, T. Oliphant, P. Peterson, et al., "SciPy: Open source scientific tools for Python", 2001.
12. M. Suzuki, *General Decomposition Theory of Ordered Exponentials*, Proc. Japan Acad. B, **69**, no. 7, 161–166, 1993.
13. H. De Raedt and K. Michielsen, *Computational Methods for Simulating Quantum Computers*, in: Quantum and Molecular Computing, Quantum Simulations, M. Rieth and W. Schommers, (Eds.), vol. 3 of *Handbook of Theoretical and Computational Nanotechnology*, 1–48, American Scientific, Los Angeles, 2006.
14. G. E. Crooks, *Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem*, arXiv:1811.08419, 2018.
15. F. G. S. L. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven, *For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances*, 2018, arXiv:1812.04170.
16. M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, *Benchmarking the quantum approximate optimization algorithm*, Quantum Inf. Process., **19**, 197, 2020.
17. 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*, Phys. Rev. X, **10**, 021067, Jun 2020.
18. P. Vikstål, M. Grönkvist, M. Svensson, M. Andersson, G. Johansson, and G. Ferrini, *Applying the Quantum Approximate Optimization Algorithm to the Tail-Assignment Problem*, Phys. Rev. Applied, **14**, 034009, Sep 2020.
19. M. Grönkvist, *The Tail Assignment Problem*, PhD thesis, Chalmers University of Technology and Göteborg University, 2005.
20. D. Willsch, M. Willsch, C. D. Gonzalez Calaza, F. Jin, H. De Raedt, M. Svensson, and K. Michielsen, *Benchmarking Advantage and D-Wave 2000Q quantum annealers with exact cover problems*, Quantum Inf. Process., **21**, 141, 2022.
21. L. Martins, *Applying Quantum Annealing to the Tail Assignment Problem*, PhD thesis, University of Porto, 2020.
22. M. Svensson, M. Andersson, M. Grönkvist, P. Vikstål, D. Dubhashi, G. Ferrini, and G. Johansson, *A Heuristic Method to solve large-scale Integer Linear Programs by combining Branch-and-Price with a Quantum Algorithm*, arXiv:2103.15433, 2021.
23. R. M. Karp, *Reducibility among Combinatorial Problems*, 85–103, Springer US, Boston, MA, 1972.
24. 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*, Nature Communications, **5**, no. 1, 4213, 2014.
25. M. S. Jattana, F. Jin, H. De Raedt, and K. Michielsen, *Assessment of the variational quantum eigensolver: application to the Heisenberg model*, Frontiers in Physics, **10**, May 2022, doi:10.3389/fphy.2022.907160.
26. 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, 51–67. Springer Netherlands, Dordrecht, 1994.

27. D. Kraft, *A software package for sequential quadratic programming*, DLR German Aerospace Center – Institute for Flight Mechanics, 1988.
28. M. S. Jattana, F. Jin, H. De Raedt, and K. Michielsen, *Improved variational quantum eigensolver via quasi-dynamical evolution*, 2022, arXiv: 2202.10130.
29. J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, *Barren plateaus in quantum neural network training landscapes*, Nature Communications, **9**, no. 1, 1–6, Dec 2018.