Quantum Software Patterns Report

This document is an auto-generated report of the quantum software patterns sourced from the PlanQK Pattern Atlas.

Pre-Trained Feature Extractor

Also known as: *Quantum Transfer Learning*

Intent

How to process large data items through quantum neural networks (QNNs) when the number of available qubits is lower than the size of a data item?

Context

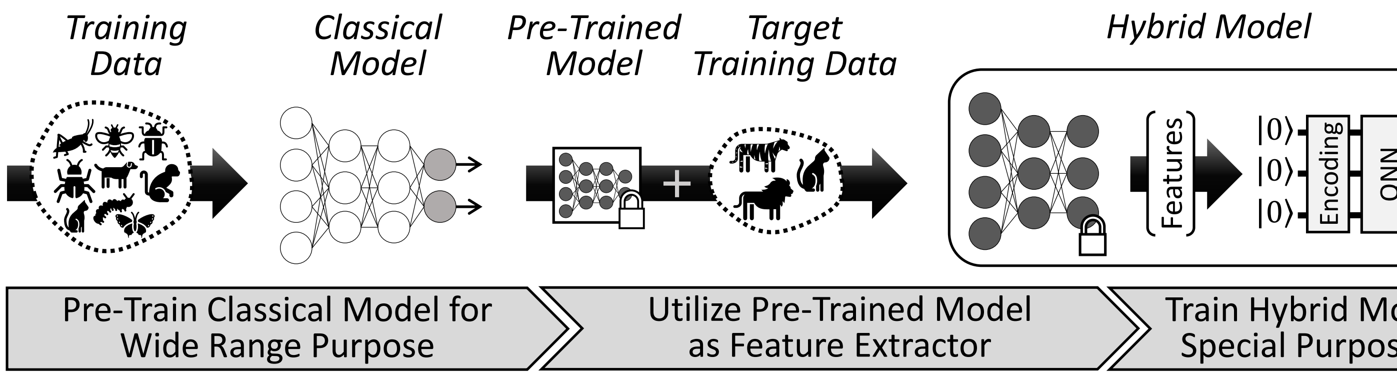
A QNN shall be trained for a specific task, that requires the processing of large data items, e.g., images or multi-dimensional data. However, the number of qubits required to load such data items into the QNN is larger than the number of qubits of the available quantum devices.

Problem & Forces

The width of circuits implementing QNNs is limited by the number of available qubits. In addition, quantum devices are scarce resources that should be utilized as efficiently as possible. However, naively reducing the original data items may result in the loss of information relevant for the computation. Large pre-trained classical models for various general tasks, such as object recognition for images, are widely available or can be created at low cost.

Solution

Use a pre-trained classical model to reduce the dimensions of the data items and train the QNN based on the reduced data. As shown in the solution sketch below, a pre-trained classical model for a wide range purpose, such as a neural network trained for object recognition, can be utilized for a hybrid QNN to be trained for a related special purpose task. Intermediate values of inputs processed through such models, e.g., those present at a condensed next-to-last neural network layer, can be seen as a compressed representation of the original data exhibiting its most significant features. Thus, the pre-trained model serves as a feature extractor. These features can be encoded into a quantum state to train the QNN for the target task.



Resulting Context

Due to the compressed representation obtained from the pre-trained feature extractor, fewer qubits are required to process data in the QNN. Furthermore, the compressed nature of the data may reduce the QNN's training time, as irrelevant information has already been omitted from the training data.

Variational Parameter Transfer

Intent

How to obtain a problem-aware parameter initialization for Variational Quantum Algorithms (VQAs) that reduces the optimization runtime?

Context

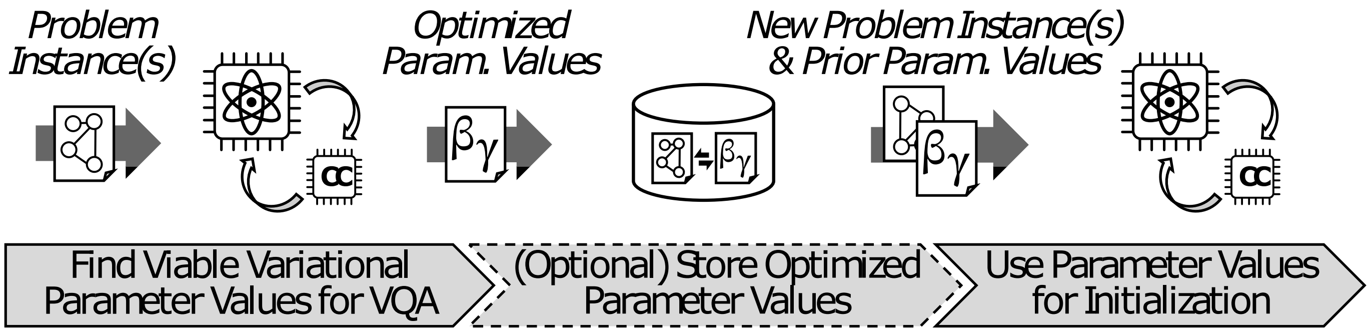
A VQA needs to be executed on a quantum device, which encompasses the optimization of its variational parameters. Parameter optimization requires repeated access to the quantum device, typically starting with random initial parameter values [Kulshrestha and Safro 2022], to sample solutions and determine a direction for their optimization, e.g., through gradient descent.

Problem & Forces

Obtaining viable parameter initializations for VQAs is challenging due to large parameter spaces and effects, such as barren plateaus [Cerezo et al. 2021] and non-convex optimization landscapes [Huembeli and Dauphin 2021]. Barren plateaus are areas with vanishing gradients in a cost function's parameter space that must be avoided, whereas local minima in non-convex optimization landscapes pose an additional challenge to efficient parameter initialization as they disturb the search for a global optimum.

Solution

Transfer viable variational parameter values from related problem instances. As shown in the solution sketch below, optimized parameter values may be stored or directly reused for new problem instances. In many cases, it can be expected that optimized parameter values for a solved problem instance are in proximity of viable parameter values for a related or similar new problem instance. Therefore, optimized parameter values from earlier executions may be utilized for a problem-aware parameter initialization instead of a random initialization. Appropriate databases, toolkits, and provenance systems for quantum computing [Shaydulin et al. 2021] [Weder et al. 2021] facilitate the optional storage of optimized parameter values for their utilization in later executions.


Problem Instance(s)
Optimized Param. Values
New Problem Instance(s) & Prior Param. Values
β_γ
CC
Find Viable Variational Parameter Values for VQA
(Optional) Store Optimized Parameter Values
Use Parameter Values for Initialization

Resulting Context

Parameter transfers can reduce the number of iterations of the optimization loop. A favorable parameter initialization can also increase the likelihood of finding globally optimal parameter values and thus increase the solution quality.

Chained Optimization

Intent

How to avoid local optima and improve convergence when optimizing variational parameter values for Variational Quantum Algorithms (VQAs)?

Context

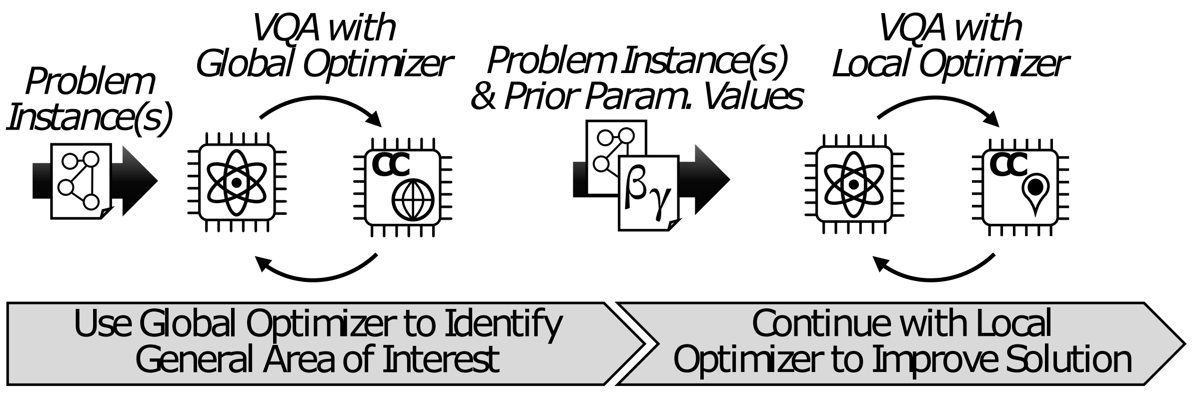
Optimal variational parameter values for a VQA need to be determined. The performance of the algorithm depends heavily on these values and a global optimum in the parameter space is needed to obtain optimal solutions.

Problem & Forces

Local minima in non-convex optimization landscapes and barren plateaus hinder the optimization, as the optimizer may be unable to reach a global optimum. Moreover, evaluating all possible parameter values is infeasibly expensive.

Solution

Chain different optimizers with different scopes or strengths together. As indicated in the solution sketch below, a global optimization strategy can be combined with a subsequent local optimizer. The former would determine a general area of interest in the overall optimization landscape. Afterward, the local optimizer is started from a point in this area of interest and searches on a smaller scale, aiming to find the global optimum.


Problem Instance(s)
VQA with Global Optimizer
Problem Instance(s) & Prior Param. Values
VQA with Local Optimizer
Use Global Optimizer to Identify General Area of Interest
Continue with Local Optimizer to Improve Solution

Resulting Context

By chaining optimizers, the subsequent optimizers utilize previously obtained results as starting points to improve upon. Thereby, optimizers are combined to benefit from their respective strengths and achieve cost-efficient optimization.

Quantum Clustering

Intent

How to partition a data set into different clusters based on their similarity utilizing a quantum device?

Context

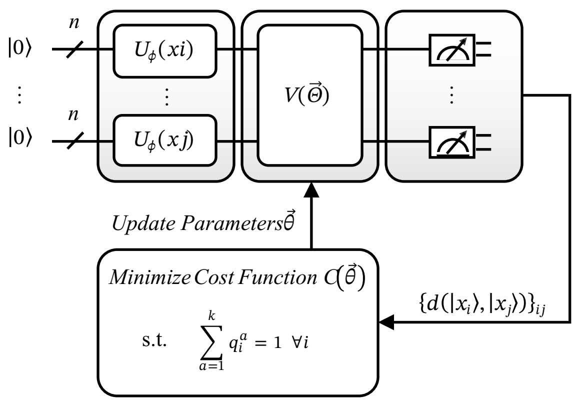
A set of unlabeled data needs to be grouped into different clusters. The clusters should organize the data points according to identified similarities.

Problem & Forces

Data sets may exhibit non-linear separability, which increases the complexity when clustering. Moreover, data sets utilized in machine learning continuously grow in size, leading to increased training times [Achiam et al., 2023]. Therefore, algorithms whose runtime scales well with the number of data points in the data set and the dimensions of the feature space are required. In addition to the general machine learning forces, also quantum-specific forces have to be taken into account. For example, loading large data sets consisting of many tuples into current quantum devices is difficult due to the high circuit depth of the required state preparation routines [Akshay et al., 2020] [Preskill, 2018].

Solution

Figure 2 gives an overview of the general clustering process. Use a quantum device to cluster the m data points $\newcommand{\state}[1]{{\left| #1 \right>}}{\state{x_i}}^m_{i=1}$ of a data set. First, the classical data points are encoded into quantum states $\newcommand{\state}[1]{{\left| #1 \right>}}{\state{x_i}}^m_{i=1}$ by applying a unitary transformation $U_ϕ$, enabling the quantum computer to process the data. Once the data points are encoded, a given ansatz $V(\vec{θ})$ is used to calculate the similarity between data points. The ansatz is a parameterized quantum circuit designed to approximate the quantum state that captures the relevant features of the data for similarity measurement. The ansatz computes the similarity either between pairwise data points or between all data points, depending on its structure [Poggiali et al., 2024]. The cost function used in this approach is designed to assign similar points to the same cluster and points with low similarity to different clusters. In the cost function, this is represented by a penalty term that penalizes distant points that are assigned to the same cluster. Additionally, the clustering process is controlled by adding constraints. To ensure that each data point is assigned to exactly one cluster, the following condition must hold $\sum^{k}_{a=1} q^a_i = 1$. Thereby, classical variables q^a_i are introduced to denote whether a data point x_i is assigned to cluster a. The quantum circuit parameters are updated iteratively to minimize the cost function.


$|0\rangle$
n
$U_\phi(xi)$
$|0\rangle$
n
$U_\phi(xj)$
$V(\vec{\Theta})$
Update Parameters$\vec{\theta}$
Minimize Cost Function $C(\vec{\theta})$
s.t. $\sum_{a=1}^{k} q_i^a = 1 \ \forall i$
$\{d(|x_i\rangle, |x_j\rangle)\}_{ij}$

Resulting Context

Utilizing a quantum clustering algorithm may enable identifying clusters in a data set exponentially faster than with a classical clustering algorithm [Khan et al., 2019c].

Often, the computational advantage of quantum clustering algorithms relies on the availability of the input data in a suitable format. Once an implementation of Quantum Random Access Memory (QRAM) Encoding is available, data can be encoded efficiently, enabling the full potential of quantum clustering.

Biased Initial State

Intent

How to utilize efficient approximations in quantum algorithms to improve the solution quality or speed up the computation?

Context

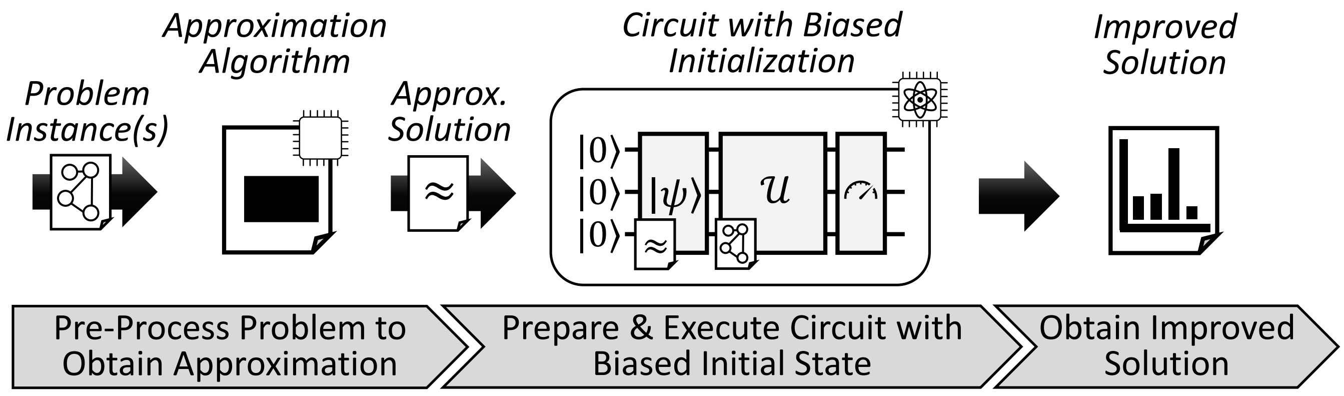
For many computationally hard problems, efficient approximation algorithms exist. However, typical quantum algorithms neglect these approximations and valuable information remains unused as the quantum algorithm starts from a neutral position. As a result, deep quantum circuits may be required, which increases accumulative error rates, and more quantum resources may be required to solve a problem.

Problem & Forces

Moreover, current quantum devices are error-prone, thus, the depth of executable quantum circuits is limited. However, including approximations requires special care, as it can limit the quantum algorithm in an unintended way [Cain et al. 2022][Truger et al. 2024]. Also, changing the initial state may require additional adaptations of corresponding parts of the quantum circuit [Egger et al. 2021][Tate et al. 2023a].

Solution

Encode approximations into the initial state of quantum circuits, thereby biasing the initial quantum state towards viable solutions. Hence, a chain of algorithm executions as depicted in the solution sketch below is beneficial: First, an efficient algorithm is utilized to approximate a solution of a given problem instance. This can often be achieved at low cost on classical hardware. Then, the initial state $|\psi\rangle$ of the subsequent quantum algorithm is biased toward the approximation and the algorithm is executed on a quantum device to obtain an improved solution.



Resulting Context

The quantum algorithm employed in the second step utilizes the approximation as a starting point to improve upon. Due to the biased initial state, optimal solutions can be explored quicker and the solution quality achievable in a set amount of time may therefore increase. Moreover, this way the workload of the overall computation can be distributed to multiple devices, e.g., classical and quantum devices.

Circuit Cutting

Also known as: *Enter your input for this section here.*

Intent

How to partition the computation of a quantum circuit into multiple smaller computations fitting the capabilities of available quantum devices?

Context

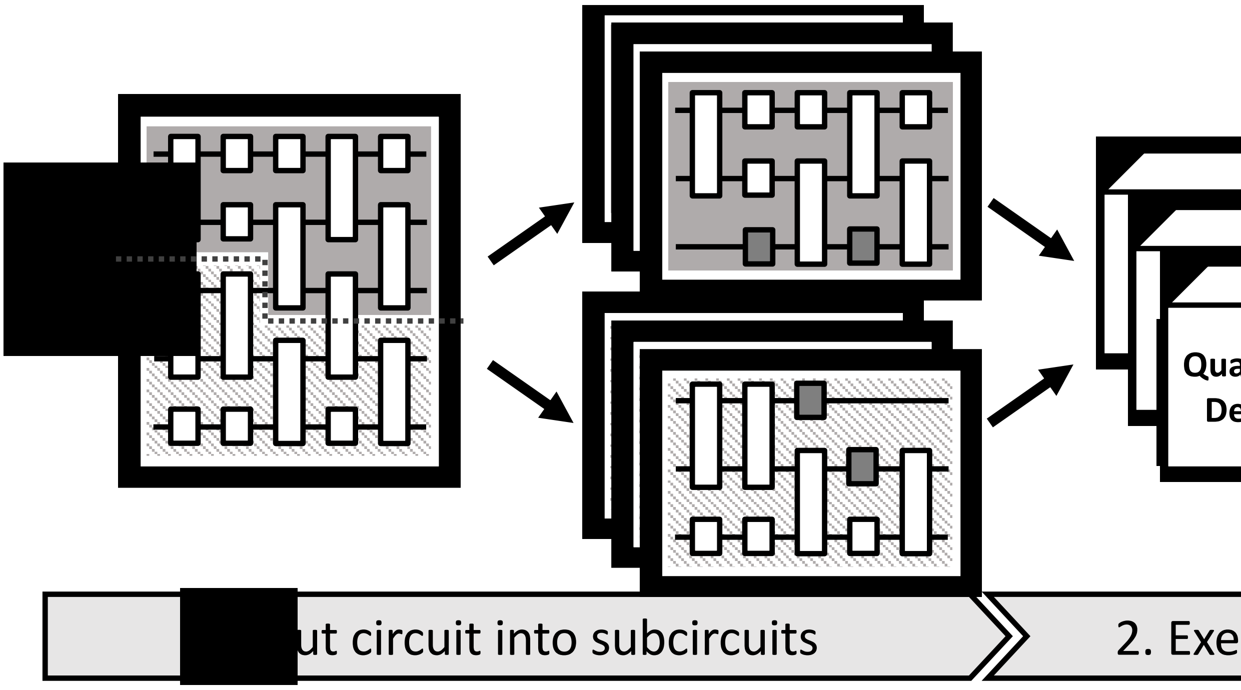
The maximum circuit width that a quantum device is able to execute is determined by its qubit count. As a result, devices with a limited number of qubits are restricted to executing circuits of small widths. Additionally, current NISQ devices face further limitations, such as high error rates and low coherence times, which impose restrictions on the number of gates that can be executed successfully in a circuit.

Problem & Forces

For smaller quantum devices, limited by their number of qubits and successfully executable gates, to contribute to the computation of a larger quantum circuit, it is necessary to divide the larger circuit into several smaller computations. Each of these smaller computations must require fewer resources, while collectively they must preserve the original circuit's result. However, several factors hinder the partitioning of a quantum circuit into multiple smaller computations. First, multi-qubit gates in a quantum circuit can create entanglement between the qubits on which they operate, resulting in the interdependence of their states. As a result, a quantum circuit with all qubits interconnected by multi-qubit gates cannot be partitioned into smaller, disconnected circuits that can be independently computed and then combined to produce the same result as the original circuit. Furthermore, removing multi-qubit gates to partition the circuit alters the computation's result. Moreover, the absence of shared entanglement or other means of quantum communication between quantum devices renders the distribution of a quantum circuit computation over multiple devices impossible.

Solution

Employ circuit cutting to divide a quantum circuit's computation into the computation of smaller circuits [Peng et al., 2020 Mitarai et al., 2021] as shown in the sketch. In the first step, the circuit is divided into multiple different variations of it, known as *subcircuits*. Each of these subcircuits can be partitioned along a cutting line, allowing the individual execution of its disconnected parts in the second step. In the final third step, these separate results can be combined for each subcircuit, and their outcomes are then merged to obtain the original circuit's output using classical post-processing.



Resulting Context

Each individual subcircuit execution requires fewer qubits and gates, consequently reducing the hardware requirements of the quantum devices and enhancing the overall computation's robustness against errors and decoherence. Replacing the original circuit with a linear combination of subcircuits resulting from the cut enables replicating the effect of entanglement and, consequently, also the result. Each partitioned subcircuit can be executed successively on one or concurrently on multiple quantum devices to decrease the overall runtime of the computation [Bravyi et al, 2022]. However, compared to directly executing the original circuit, more shots are needed due to the multiplicative factor each cut introduces for estimating the expectation value with desired statistical accuracy [Piveteau et al, 2024]. To minimize the resulting additional overhead and error propagation, the placement of cuts, e.g. WIRE CUTs and GATE CUTs, and the structure of the subcircuits should be optimized [Casciola et al., 2022]. Achieving this optimization can be automated through the utilization of mixed-integer programming techniques [Tang et al., 2021].

Uniform Superposition

Intent

Create a uniform superposition of all possible states of a quantum register

Context

The power of quantum algorithms partially originates in the ability to represent multiple states at once, the so-called quantum parallelism.
To realize quantum parallelism, some qubits of a quantum register must be brought into superposition.
Many algorithms start with obtaining an equally weighted superposition in (a part of) a quantum register. Therefore, for every state represented by (this part of) the quantum register, the measurement probability is the same.

Problem & Forces

Not available

Solution

After initializing the quantum register as the unit vector $\left|0 \ldots 0\right>$, a Uniform Superposition is created via the Hadamard transformation:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} H^{\otimes n} \left(\left| 0 \right>^{\otimes n} \right)= \dfrac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x \right>$$ The quantum register may also include ancilla bits that must not necessarily be brought into superposition. For example, if the last part of the quantum register is used for ancilla qubits, a different Initialization on the two parts of the register can be accomplished by using a tensor product operator $H^{\otimes n}\otimes U$ where the Hadamard transformation $H^{\otimes n}$ only operates on the first part and U operates on the ancilla qubits. If the ancilla qubits are not brought into superposition, U is the Identity (e.g., $U=I^{\otimes m}$): $$ \newcommand{\state}[1]{{\left| #1 \right>}} H^{\otimes n} \otimes I^{\otimes m}

\left(\left| 0 \right>^{\otimes n} \otimes \left| 0 \right>^{\otimes m} \right)= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x \right> \otimes \state{0}^{\otimes m} $$

Resulting Context

A Uniform Superposition can be achieved by performing a single parallel operation on each qubit, increasing the depth of the entire circuit by only one.

Matrix Encoding

Also known as: *Dynamic Encoding [Schuld and Petruccione 2018]*

Intent

Represent a matrix as an operation on a quantum computer

Context

A matrix A has to be represented on a quantum computer as an operation U. Therefore, it is not sufficient to represent the entries of the matrix by a data encoding pattern as this does not define an operation.

Problem & Forces

Not available

Solution

If A is not Hermitian, encode $$H = \left(\begin{matrix}0 & A\ A^{\dagger} & 0 \end{matrix} \right) $$ instead of A. Since H is hermitian, $U=e^{-iHt}$ is unitary and can be used to encode H where t specifies the time for which the operation is applied (usually t is chosen to be small).

Resulting Context

Since the eigenvectors $\newcommand{\state}[1]{{\left| #1 \right>}}{\state{v_i}}$ of the $n \times n$ matrix H form a basis, each state vector $\ket{\psi}$ can be written as a linear combination of these eigenvectors: $$ \newcommand{\state}[1]{{\left| #1 \right>}}\state{\psi}= \gamma_1 \state{v_1} + \ldots + \gamma_n \state{v_n}$$ Applying U to this state results in the following state [Schuld and Petruccione 2018]: $$\newcommand{\state}[1]{{\left| #1 \right>}}e^{-iHt} \state{\psi} = e^{-i \lambda_1 t} \gamma_1 \state{v_1} + \ldots + e^{-i \lambda_n t} \gamma_n \state{v_n}$$ If $\ket{\psi}$ is an eigenvector of H and thus, only one $\gamma_i \neq 0$, the state acquires a global phase shift: $U \gamma_i \ket{v_i} = e^{-i \lambda_i t} \gamma_i \ket{v_i}$ (see original paper for a definition of phase shift). This nicely reflects that applying the matrix A to an eigenstate leads to a scaling factor. The main drawback of this encoding is that the depth of the circuit that implements U can be exponential.

Schmidt Decomposition

Intent

Prepare an arbitrary state

Context

A state $\newcommand{\state}[1]{{\left| #1 \right>}}\state{s}$ has to be prepared on an empty n-qubit register. If no state preparation method is known that exploits the structure of this state to prepare it efficiently, a method for creating an arbitrary state can be used instead.

Problem & Forces

Not available

Solution

To generate a circuit for the creation of $\newcommand{\state}[1]{{\left| #1 \right>}}\state{s}$, it first needs to be expressed in terms of two subspaces V and W that span $H^{\otimes n}$. First, orthogonal basis ${f_1, \ldots, f_k}\in V$ and ${g_1, \ldots, g_k}\in W$ are chosen and $\ket{s}$ is represented as a linear combination of these basis vectors: $$\newcommand{\state}[1]{{\left| #1 \right>}}\state{s}=\sum_{i, j} b_{ij} \cdot f_i \otimes g_j $$ Then, the singular value decomposition (SVD) of the matrix ${M = { b_{ij} }}$ is computed (see [Olver et al. 2006] for detailed instructions): $$M= \left(\begin{matrix}U_1 U_2 \end{matrix} \right) \left(\begin{matrix}A \ 0 \end{matrix} \right) V^* $$ where the matrix U obtained by the SVD is rewritten by U_1 and U_2. The entries of the diagonal matrix A build the set ${\alpha_1, \ldots \alpha_m}$ which defines the *Schmidt decomposition* of $ \newcommand{\state}[1]{{\left| #1 \right>}}\state{s}$: $$\newcommand{\state}[1]{{\left| #1 \right>}}\state{s}=\sum_{i=1}^{m} \alpha_{i} \cdot u_i \otimes v_i, \alpha_{i} \in \mathbb{R} \geq 0, \text{where} \sum_{i=1}^{m} \alpha_{i} = 1 $$ where $ \alpha_{1}, \ldots ,\alpha_{m}$ are the *Schmidt coefficients* for the *Schmidt basis* ${u_i}$, ${v_i}$. The circuit in the pattern sketch can be used to prepare $ \newcommand{\state}[1]{{\left| #1 \right>}}\state{s}$ on an empty register [Abhijith et al. 2018]: First, B transforms the amplitude of the first register to the Schmidt coefficients. Then, a series of CNOT operations copies this state to the second register. Finally, U_1 and V transform the computational basis states ${e_i}$ into the Schmidt basis states: $$(U_1 \otimes V) \sum_{i=1}^{m} \alpha_i \cdot e_i \otimes$$ For the execution on a quantum computer, the unitary matrices must be further decomposed into one and two qubit gates.

Resulting Context

The state $\newcommand{\state}[1]{{\left| #1 \right>}}\state{s}$ is created in the register. For this state, the Schmidt coefficients α_i are known which can be used to quantify entanglement [Nielsen and Chuang 2002]. The state $ \newcommand{\state}[1]{{\left| #1 \right>}}\state{s}$ is separable if and only if exactly one of the Schmidt coefficients is non-zero. In the worst case, the depth of the circuit is exponential (more precisely: $\frac{23}{48} 2^n$ [Plesch and Brukner 2011]).Note that arbitrary state preparation was shown to be of exponential complexity, i.e., a circuit of exponential depth will always be needed in the worst case.

Uncompute

Also known as: *This pattern has also been referred to as Unentangling or Copy-Uncompute.*

Intent

Remove entanglement that resulted from a previous computation

Context

Quantum algorithms often use ancilla qubits as temporary qubits for their computations. After a computation, these qubits are often still entangled with the computational basis of the quantum register.
This prevents unrestricted access to the results of the computation.
This is especially problematic if the performed calculations are only intermediate steps within a larger algorithm. For example, assume a computation should produce a weighted superposition $\newcommand{\state}[1]{{\left| #1 \right>}} \sum \alpha_i \state{\phi_i}$, but produces $\newcommand{\state}[1]{{\left| #1 \right>}} \sum \alpha_i \state{\phi_i} \state{\psi_i}$ instead, where $\newcommand{\state}[1]{{\left| #1 \right>}} \state{\psi_i}$ denotes the state of the ancilla qubits. The second part of the register containing the ancilla qubits cannot be discarded unless

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \sum \alpha_i \state{\phi_i} \state{\psi_i} = \left(\sum \alpha_i \right) \state{\phi_i} \state{\psi_i}$$ holds, i.e., unless the two parts of the register are separable.

Problem & Forces

Not available

Solution

When computing a function f, many algorithms map $\newcommand{\state}[1] {{\left| #1 \right>}} \state{x}\state{0}\state{0}$ to $\newcommand{\state}[1]{{\left| #1 \right>}} \state{x}\state{g(x)}\state{f(x)}$ (Derovic et al. 2018).
As a result, the second part of the register represents a workspace containing the byproduct $g(x)$ of the computation of $f(x)$, which is not needed anymore.
This garbage part of the register has to be reset, especially if the following parts of the algorithm expect a proper initialization of the workspace as $ \newcommand{\state}[1]{{\left| #1 \right>}} \state{0 \ldots 0}$.

More specifically, assume the following state to be the result of the computation U_f:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \state{x}\state{0}\state{0} \xrightarrow{U_f} \sum \alpha_y \state{x}\state{y} \state{f(x)}$$

with $\newcommand{\state}[1]{{\left| #1 \right>}} \state{g(x)} = \sum \alpha_y \state{y}$ being the garbage state. Then, a fourth register is added and initialized as $\newcommand{\state}[1]{{\left| #1 \right>}} \state{0}$. Then, CNOT is applied (bitwise) to this fourth register controlled by the third register containing the actual results of the computation. Thereby, $f(x)$ is copied to the fourth register which results in $\newcommand{\state}[1]{{\left| #1 \right>}} \sum \alpha_y \state{x} \state{y} \state{f(x)} \state{f(x)}$.
To reset the potentially entangled first three registers, the inverse operator U^{-1}_f is applied to them, resulting in $\newcommand{\state}[1]{{\left| #1 \right>}} \state{x} \state{0} \state{0} \state{f(x)}$. By application of the SWAP operator, the entries of the last two registers are swapped: $\newcommand{\state}[1]{{\left| #1 \right>}} \state{x} \state{0} \state{f(x)} \state{0}$. The fourth register is in the state $ \newcommand{\state}[1]{{\left| #1 \right>}} \state{0}$ again and can be discarded, leaving $\newcommand{\state}[1]{{\left| #1 \right>}} \state{x} \state{0} \state{f(x)}$ as final result (see (Derovic et al. 2018) for a more detailed description). How a Uncompute operation can be realized in several other situations is described in (Proos and Zalka 2003).

Resulting Context

The resulting register $\newcommand{\state}[1]{{\left| #1 \right>}} \state{x} \state{0} \state{f(x)}$ contains the input x and the computed function values $f(x)$.
The previous entanglement (caused by the computation) is no longer present.

Basis Encoding

Also known as: *Enter your input for this section here.*

Intent

Represent data elements in a quantum computer in order to perform calculations

Context

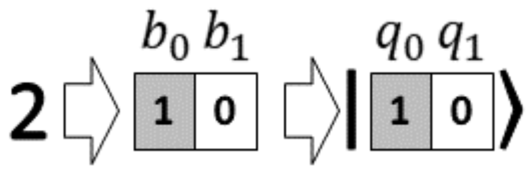
A quantum algorithm requires numerical input data X for further calculations

Problem & Forces

Not available

Solution

The main idea for this encoding is to use the computational basis $|0...00\rangle, |0...01\rangle, \ldots, |1...11\rangle$ to encode the input data: An input number x is approximated by a binary format $x := b_{n-1}\ldots b_1 b_0$ which is then turned into the corresponding basis vector $|x \rangle:=|b_{n-1} \ldots b_1 b_0\rangle$. For example, the number \"2\" is represented as 10 which is then encoded by $|10\rangle$ (see sketch). In general, this leads to the following encoding: $X \approx \sum_{i=-k}^m b_{i} 2^i \mapsto | b_m \ldots b_{-k} \rangle$ where X is first approximated with a precision of k significant digits and then represented by a basis vector.



Pattern sketch, taken from: Weigold, Manuela; Barzen, Johanna; Leymann, Frank; Salm, Marie: Data Encoding Patterns for Quantum Algorithms. In: The Hillside Group (Hrsg): Proceedings of the 27th Conference on Pattern Languages of Programs (PLoP '20).

Resulting Context

This encoding can be categorized as digital encoding because it is suitable for arithmetic computations Leymann and Barzen 2020. For input numbers which are approximated by l digits, l qubits are needed for its representation. To realize this encoding, the initial $|0\rangle$ state of qubits that represent a ’1’ digit must be flipped into $|1\rangle$. For one qubit, this can be done by a single operation, and thus, this encoding can be prepared in linear time.

Angle Encoding

Also known as: *This pattern has also been referred to as Qubit Encoding (LaRose and Coyle 2020) since each qubit represents a single data point. The resulting encoding of this pattern is not entangled, thus, another alias for this pattern is (Tensor) Product Encoding (Leymann and Barzen 2020).*

Intent

"Represent each data point by a separate qubit" (Weigold et al. 2021)

Context

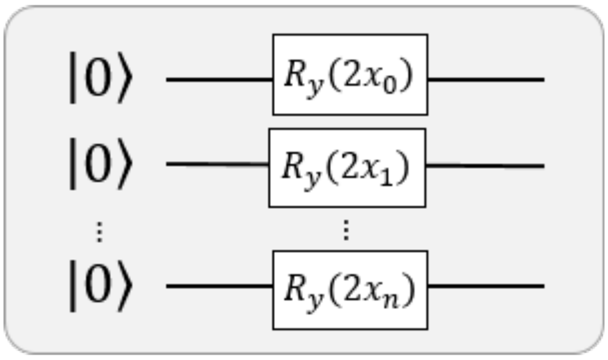
In the current NISQ era, an algorithm requires an encoding schema that is efficient in terms of operations. This enables to perform more operations within the decoherence time after encoding the data.

Problem & Forces

Not available

Solution

As a first step, each data point of the input is normalized to the interval $[0,\frac{\pi}{2}]$. To encode the data points, a rotation around the y-axis is used (see solution sketch) for which the angle depends on the value of the normalized data point.



Pattern sketch, taken from: Weigold, Manuela; Barzen, Johanna; Leymann, Frank; Salm, Marie: “Expanding Data Encoding Patterns For Quantum Algorithms.” In: 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C), IEEE, 2021.

Resulting Context

This creates the following separable state (Weigold et al. 2021):

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \state{\psi} = \bigColVec{\cos{x_0} \ \sin{x_0}} \otimes \bigColVec{\cos{x_1} \ \sin{x_1}} \otimes \ldots \otimes \bigColVec{\cos{x_n} \ \sin{x_n}} $$

It can easily be seen that one qubit is needed per data point which is not optimal. To load the data, the rotations on the qubits can be performed in parallel, thus, the depth of the circuit is optimal.

Creating Entanglement

Intent

Enforce a strong correlation between qubits by entangling them.

Context

Entanglement is a unique characteristic of quantum mechanics and one of the causes for the power of quantum algorithms (Bruß and Macchiavello 2011).
Although entanglement is not necessarily needed for a powerful quantum algorithm (Biham et al. 2004), it is required to achieve an exponential speedup over classical algorithms (Jozsa and Linden 2003).
Consequently, a quantum register is often entangled for further processing after the Initialization.

Problem & Forces

Not available

Solution

There exist numerous approaches for the creation of an entangled state. For example, entanglement can be created by a Boolean function $f:{0, 1}^n \rightarrow {0, 1}^m$, a corresponding unitary operation

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} U_f: {0, 1}^{n+m} \rightarrow {0, 1}^{n+m}, U_f \left(\left| x,y \right> \right) = \left| x,y\oplus f(x) \right>$$ and a Uniform Superposition in the first n qubits of the quantum register:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} U_f \left(H^{\otimes n} \otimes I^{\otimes m} \right) \left(\left| 0 \right>^{\otimes n} \otimes \left|0 \right> ^{\otimes m} \right) $$

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} = U_f \left(\dfrac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x \right> \otimes \state{0} ^{\otimes m} \right) $$ The resulting state is entangled. For $f = id$, $U_{f}=CNOT$

and consequently $CNOT \left(H \otimes I \right) \left(\left| 0 \right> \otimes \left|0 \right> \right)$ is entangled.

Resulting Context

Unitary transformations that create entanglement must include multi-qubit operators such as CNOT.
Such operators typically have a lower gate fidelity than single qubit operators and therefore increase the overall gate error.

Error Correction

Also known as: *Enter your input for this section here.*

Intent

How to detect and correct errors occurring during the execution of a quantum circuit?

Context

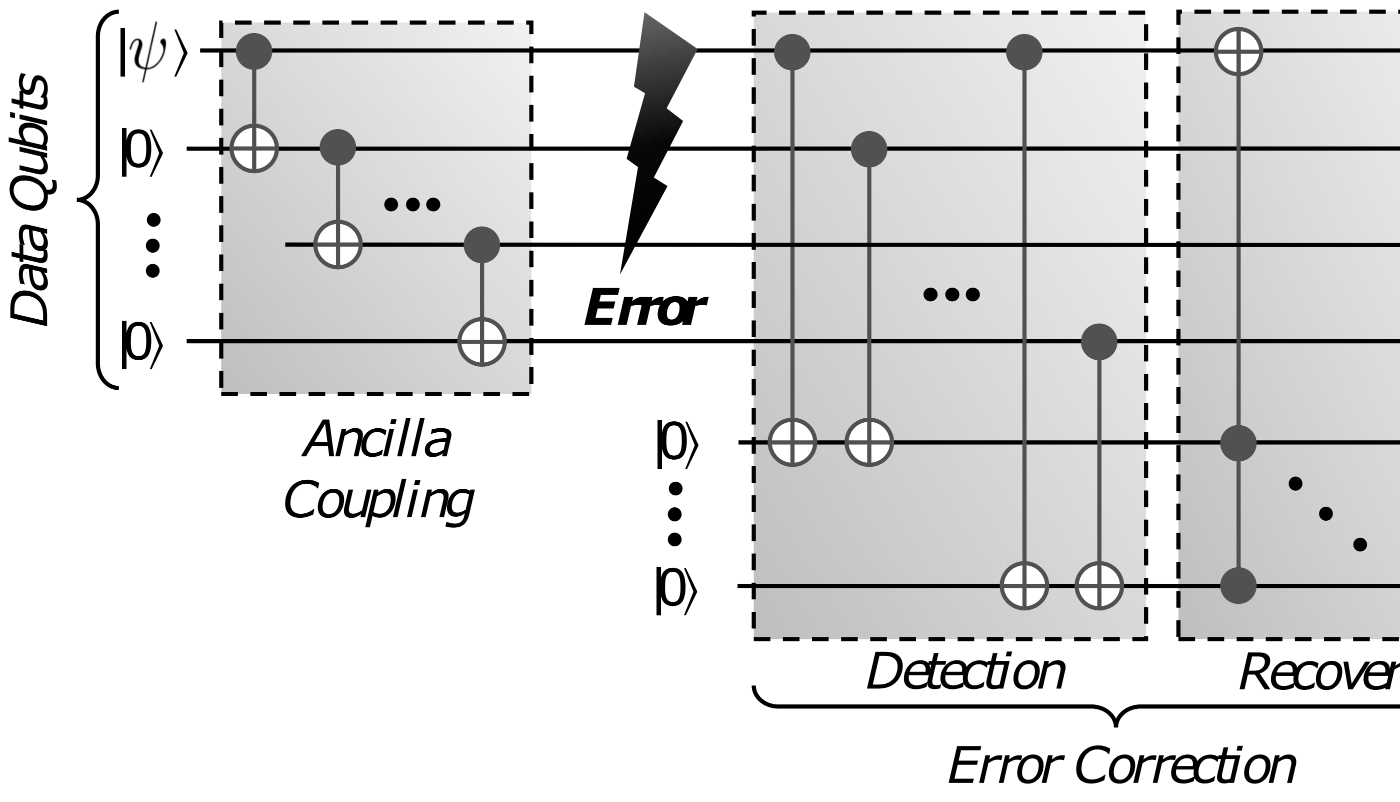
A quantum algorithm needs to be run on a quantum device. The quantum device's performance is limited by various error sources, such as gate errors and crosstalk. The prevention of these errors enables the execution of large-scale quantum algorithms for real-world problems.

Problem & Forces

Quantum devices unavoidably cause a certain amount of errors due to the fragility of coherent quantum states [Devitt et al. 2013]. Furthermore, contrary to classical bits, qubits can not be copied. Hence, classical error correction can not be used for quantum computers and new quantum-specific methods need to be developed. However, these methods can be costly in terms of quantum resources, as they require a large number of additional qubits and quantum gates. To enable scalable quantum computing for real-world problems, all kinds of errors occurring in quantum devices need to be detected and corrected. In general, the correction of errors is preferred over their mitigation, since even minor remaining post-mitigation errors slowly stack up during the computation and ultimately lead to an imprecise result.

Solution

Detect and correct quantum errors using quantum error correction codes [Devitt et al. 2013]., which are added to the executed circuit. With these correction codes, many physical qubits are combined into one logical qubit. As a result of this bundling, errors in the original qubit can be first detected and then corrected. The figure below depicts a solution sketch showcasing the general building blocks of a quantum error correction procedure. The shown instance applies an error correction code that can detect and fix bit-flip errors in the computational basis. For the correction of errors from other sources, similar processes can be applied. First, the ancilla coupling is created, by encoding the state $\ket{\psi}$ of a single physical qubit into multiple ancilla qubits. These qubits now hold the logical qubit's data and are called data qubits in the following. Next, some unitary transformation is applied to the logical qubit, possibly resulting in an error. In order to detect an error, additional ancilla qubits are employed to check the parity of the data qubits. Based on the discovered syndrome, the error-free state can be recovered in the recovery phase. Note that the process is assumed to only have errors at the unitary transformation step, which is denoted by the error indicator. Further, the number and type of detectable errors depends on the applied error correction code.



Resulting Context

When applying quantum error correction, computational errors can be prevented, enabling error-free systems of logical qubits. Thus, error correction is making fault-tolerant quantum computation feasible. The good scalability of error correction, enables the accurate execution of large algorithms

---

Gate Error Mitigation

Also known as: *Enter your input for this section here.*

Intent

How to reduce the negative impact of noisy gate executions such that the pre-measurement state is closer to the expected error-free state?

Context

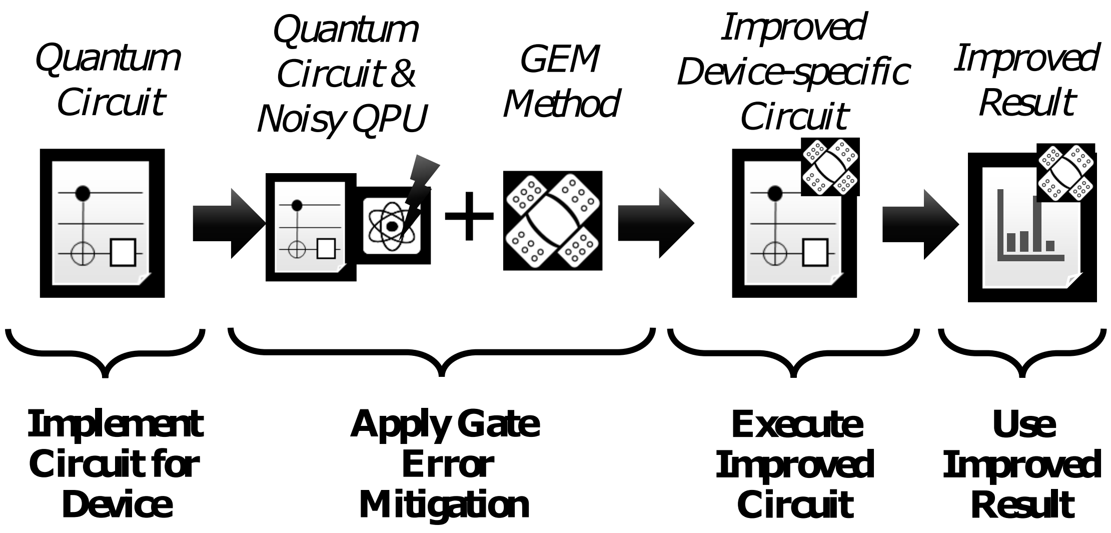
A NISQ-compatible quantum algorithm, e.g., VQE, needs to be run on a quantum device. The device's gate implementations are error-prone, causing errors in the quantum computation. To obtain precise results for the executed algorithm, the measured state needs to be computed accurately. Thus, it is crucial to mitigate the effects of gate errors.

Problem & Forces

The execution of gates on current NISQ devices is not perfectly accurate. Hence, every execution of a gate causes a minor error. These errors keep accumulating, eventually making large computations impossible. The pulses used for the implementation of gate operations can be controlled on many quantum devices. Therefore, custom pulse schedules can be used to individually calibrate gates. Furthermore, the capabilities of current quantum devices are limited, e.g., the number of qubits and the decoherence times are bound. Thus, minimal additional quantum resources, such as gates and qubits, shall be used for error mitigation.

Solution

Mitigate the impact of gate errors by applying a Gate Error Mitigation (GEM) method. The mitigation of gate errors has to be performed before the execution of the quantum circuit, as occurring errors otherwise accumulate during the computation, making it difficult to retrace them. The resulting pre-measurement quantum state is closer to the expected error-free state, therefore, providing more accurate measurement results. The figure below depicts a solution sketch for GEM. First, the circuit is implemented. Afterwards, a GEM method is applied, modifying the circuit, to generate a more precise implementation for the selected device. The circuit modifications can range from simple gate additions over custom gate pulse adjustments to full circuit rewrites based on Machine Learning (ML). Next, the improved circuit is executed on the quantum device. Finally, the improved measurement result can be evaluated to obtain a more precise solution.


Quantum Circuit
Quantum Circuit & Noisy QPU
GEM Method
Improved Device-specific Circuit
Improved Result
Implement Circuit for Device
Apply Gate Error Mitigation
Execute Improved Circuit
Use Improved Result

Resulting Context

GEM can significantly reduce the impact of errors caused by erroneous gate executions. As a consequence, the state computed by the quantum algorithm is closer to the expected error-free quantum state and a more precise algorithm result can be obtained. However, the mitigation process may induce additional quantum gates into the circuit or require classical pre-processing to calculate optimal device calibrations, e.g., gate pulse calibrations. Generally, GEM methods can be used in combination with other error mitigation methods, such as REM to reduce the overall error further.

Quantum Kernel Estimator (QKE)

Intent

Use a quantum routine to estimate a kernel for a classical SVM.

Context

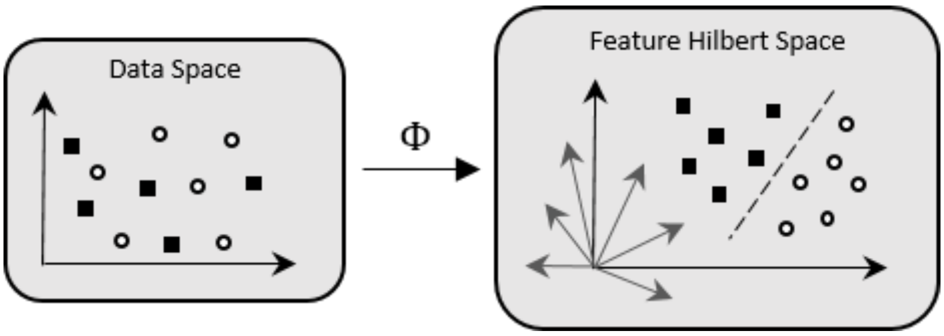
A support vector machine (SVM) must be found to classify a set of data points ${x_i} \subseteq \mathcal{R}^d$ according to their labels. Therefore, a hyperplane must be found that (i) separates the data points of the different classes and (ii) maintains a maximal distance to the data points. A large margin between the hyperplane and the data points ensures that unseen data points are classified correctly with a high probability. The given data set is not guaranteed to be linearly separable, and thus, it may not be possible to find such a separating hyperplane in the original space. If this is the case, a hyperplane in a higher dimensional feature space to which the data points are mapped (implicitly) can be used instead.

Problem & Forces

Not available

Solution

To find a separating hyperplane, a quantum computer is used to estimate the kernel function $K(x,x')=|\left<\phi(x)|\phi(x')\right>|^2$: A pair of data points (x,x') is encoded into the Hilbert Space according to a quantum feature map ϕ (see solution sketch). This allows to use, e.g., a SWAP test routine to estimate the inner product $\left<\phi(x)|\phi(x')\right>$ of the two points (Schuld and Killoran) The result can then be used to compute the kernel function for this pair of data points, based on which the SVM is optimized on a classical computer.



Pattern sketch, taken from: Weigold, Manuela; Barzen, Johanna; Leymann, Frank; and Vietz, Daniel: Patterns For Hybrid Quantum Algorithms. In: Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2021).

Resulting Context

Both the training as well as the classification is efficient, given that the evaluation of the inner products can be done in an efficient manner. The main advantage of this setup is that the quantum computer has the potential to compute inner products in a feature space that cannot be evaluated efficiently on a classical computer. Still, a key open question regarding this setup remains how to choose a feature map for a given data set.

Function Table

Also known as: *Enter your input for this section here.*

Intent

Compute a function table of a finite Boolean function

Context

A classical algorithm must evaluate a given function $f:{0, 1}^n \rightarrow {0, 1}^m$ for each value of the domain for computing a function table. By exploiting quantum parallelism, a quantum algorithm is able to compute all values of such a finite Boolean function in a single step. This is useful to speed-up algorithms that reveal global properties of a respective function f. Note that for $m=1$, a Boolean function $f:{0, 1}^n \rightarrow {0, 1}$ is often an indicator function which can be used to solve a decision problem.

Problem & Forces

Not available

Solution

For the computation of the function table, the quantum register is split into two parts: The first part consists of n qubits x which represent the domain of the Boolean function f in the computational basis.
The second part of the register contains m qubits y which will be used to represent the values of f. The unitary operator implementing the computation of the function table is then defined as:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}}

U_f \left| x,y\right>=\left| x, y\oplus f(x) \right> $$ As described in Uniform Superposition, the register is first brought in uniform superposition by initializing it as $\newcommand{\state}[1]{{\left| #1 \right>}} \state{0}^{\otimes n} \otimes \left| 0 \right>^{\otimes m}$, and applying the Hadamard transformation $H^{\otimes n}$ on the first part, leaving the second part of the register in the $ \newcommand{\state}[1]{{\left| #1 \right>}} \left| 0 \right>^{\otimes m}$ state.
Then, the operator U_f is applied only once to the complete register generating the following function table:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \state{0}^{\otimes n} \state{0}^{\otimes m} \xrightarrow{H^{\otimes n} \otimes I} \left(\frac{1}{\sqrt{2^n}} \sum_x \state{x} \right) \otimes \state{0}^{\otimes m} $$

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \xrightarrow{U_f} \frac{1}{\sqrt{2^n}} \sum_x \state{x}\state{f(x)} $$ If f is an one-dimensional function (e.g., for solving a decision problem), the register is initialized as $\newcommand{\state}[1]{{\left| #1 \right>}} \state{0}^{\otimes n} \state{1}$.
Afterward, the Hadamard operation is performed on the complete register, i.e. the operator $H^{\otimes n+1}$ is used.
Application of U_f results in the following state:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \state{0}^{\otimes n} \state{1} \xrightarrow{H^{\otimes n} \otimes H} \left(\frac{1}{\sqrt{2^n}} \sum_x \state{x} \right) \otimes \state{-}$$

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \xrightarrow{U_f} \left(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \state{x} \right) \otimes \state{-} $$

Resulting Context

For the general case of $m>1$, the register is in a superposition of all data values in the first register and their corresponding data values in the second register. In the special case of a one-dimensional decision function f, the register contains a superposition of all computational bases, where the sign indicates the outcome of the decision function: A minus sign indicates an outcome of 0 whereas a plus sign indicates an outcome of 1 - this is also referred to as "phase kickback".

Amplitude Amplification

Intent

Increase the probability of finding a solution

Context

For a specific indicator function f the corresponding function table may list all possible solutions of a problem, i.e. $f(x)=1 \Leftrightarrow x$ is a solution to the problem.
When the corresponding state is measured, a solution can be found with a certain probability. Since measuring destroys the state, the computation has to be repeated if no solution is found by the measurement in order to be able to do another measurement of the state. To keep computational costs low, a mechanism without measurements is required.

Problem & Forces

Not available

Solution

The overall state is transformed such that the probability of measuring certain values of interest increases with every iteration by modifying their amplitude (Brassard et al. 2002). This is done via the help of a phase shift S_G^π, which changes the sign of the phase of elements in the set of solutions G while all other elements remain unchanged. Another phase shift that is used is S_0^π which changes only the sign of the zero state $\newcommand{\state}[1]{{\left| #1 \right>}} \state{0}$. Suppose there is an algorithm U for computing approximate solutions (without any measurements). The following unitary operation can be defined:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} Q = -U S_0^\pi U^{-1} S_G^\pi$$ If the success probability of the algorithm U is t, the average amount of iterations required to find a solution is $1/t$. It is assumed that $\newcommand{\state}[1]{{\left| #1 \right>}}U\state{0}$ has a non-zero amplitude in G, otherwise, there cannot be a speedup. If U indeed has this property, the above-defined unitary operation Q will create a solution within $O(\sqrt{1/t})$ iterations, i.e., a quadratic speedup can be achieved. The amount of iterations to be realized by Q is about $\newcommand{\state}[1]{{\left| #1 \right>}} \frac{\pi}{4} \frac{1}{P_G U \state{0}}$, where P_G is the projection onto the subspace spanned by G.

Resulting Context

The probability of measuring a solution is increased.

Post-Selective Measurement

Intent

Select one branch of a superposition to proceed with

Context

As quantum operations are unitary, they perform linear transformations. However, sometimes a non-linear transformation is desirable. For example, if the result of a computation is stored in one branch of the superposition, e.g., $$ \newcommand{\state}[1]{{\left| #1 \right>}}\state{\psi}=\frac{1}{\sqrt{2}} \state{1}\state{f(x)} + \frac{1}{\sqrt{2}} \state{0}\state{g(x)}$$ one would like to proceed with $\newcommand{\state}[1]{{\left| #1 \right>}}\state{1}\state{f(x)}$ and discard the rest of the superposition.

Problem & Forces

Not available

Solution

Use measurement to force the quantum state to collapse into one of the two branches. For the example above, measuring the first qubit in the computational basis results in either $\newcommand{\state}[1]{{\left| #1 \right>}}\state{0}$ or $\newcommand{\state}[1]{{\left| #1 \right>}}\state{1}$. If the measurement indicates that the preferred branch was selected (i.e., $\newcommand{\state}[1]{{\left| #1 \right>}}\state{1}$ for our example)), one can proceed with further calculations (shown in the pattern sketch). Otherwise, the current computation is discarded and restarted from the beginning, i.e., by first resetting each qubit to $\newcommand{\state}[1]{{\left| #1 \right>}}\state{0}$ followed by the operations of the rest of the circuit.

Resulting Context

As the resulting approach is probabilistic, thus, the average number of iterations needed depends on the amplitude of the selected branch. In the example above, the algorithm proceeds with a probability of 50\%.

Initialization

Also known as: *This pattern has also been referred to as State Preparation.*

Intent

Initialize the input of a quantum register, taking into account the prerequisites of the subsequent steps of the algorithm.

Context

Usually, the underlying problem to be solved by a quantum algorithm is represented by specific parameters. These parameters must be given as input data to the algorithm in order to solve the problem. In most algorithms, the process of loading the input data is part of the quantum algorithm itself, which is defined as a unitary transformation U and measurements.
In this case, the overall algorithm $U = U_n \circ \ldots \circ U_{i} \circ U_{i-1} \circ \ldots \circ U_1,$ can be split up into two parts. The operators of the first part $U_1, \ldots , U_{i-1}$ encode the input data into the quantum register according to a defined encoding, whereas the operators of the second part $U_i, \ldots , U_n$ are used to solve the problem.
Since $U_{i-1} \circ \ldots \circ U_1$ set the register to an initial state, this step is referred to as state preparation.

Problem & Forces

Not available

Solution

Frequently, the unit vector $\left| 0 \ldots 0\right>$ is used as initialization of a quantum register.
Some qubits of the register can be used as so-called ancilla bits (working qubits) which may be used for the storage of intermediate results or quantum error correction. For example, to compute the function table of a Boolean function $f:{0, 1}^n \rightarrow {0, 1}^m$, the overall register is initialized as $\left| 0 \right>^{\otimes n}\left| 0 \right>^{\otimes m}$ (including m ancilla bits in the second part of the register).
To expose membership in an indicator function-based set (e.g., in decision problems) often an initialization with $\left| 0 \right>^{\otimes n}\left| 1 \right>$ is chosen.
The membership to the set is then indicated by changing the sign of the qubits representing members of this set.

Resulting Context

More advanced states may be prepared which build on the previously described initialization techniques. For example, in (Cortese and Braje 2018) various algorithms for loading classical bits into a quantum register are presented.
Complex vectors can be loaded as described in (Nielsen and Chuang 2002). (Derovic et al. 2018) describes how a real-valued vector can be loaded; therefore, it is also possible to load a matrix that is represented as a set of vectors (Kerenidis and Prakash 2016).

Unified Observability

Also known as: –

Intent

How to ensure reproducibility, understandability, and quality when executing hybrid quantum applications?

Context

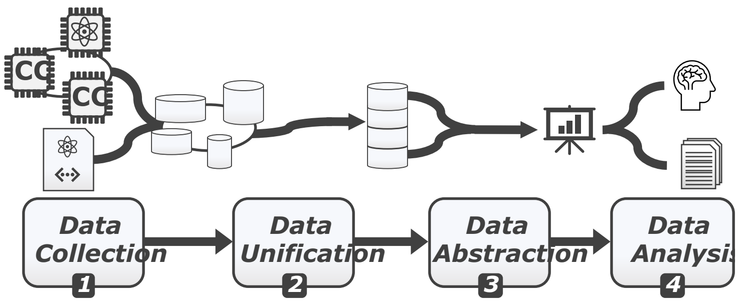
Quantum applications comprise a multitude of classical programs and quantum circuits which are typically executed in a heterogeneous execution environment, e.g., utilizing classical and quantum cloud offerings [Beisel et al., 2024], [Leymann et al., 2020].

Problem & Forces

When executing hybrid quantum applications in heterogeneous execution environments, collecting all necessary data is difficult: The data must be gathered using the different APIs or SDKs of the utilized quantum and classical cloud offerings, which often change, e.g., when new features are released. Furthermore, the offerings might not provide all the required data Weder et al., 2021a. Finally, some of the data also changes over time, e.g., the qubit decoherence times or the error rates [Tannu and Qureshi, 2019]. Analyzing the data is complicated by different data formats and abstraction levels of the provided data [Beisel et al., 2024]. Moreover, hybrid quantum applications are typically developed and operated by interdisciplinary teams with various backgrounds, e.g., physics, mathematics, and software engineering, requiring different information [Weder et al., 2022].

Solution

Figure 6 shows the phases required to achieve unified observability. Data about the execution of the quantum application and the used quantum and classical resources must be collected continuously. Persistently store these data using a provenance system that automatically unifies data using transformation methods. For example, quantum cloud offerings use both the fidelity and error rate metrics to describe the quality of their gate operations, where Error Rate = 1−Fidelity. Use benchmarks to retrieve data that is not provided by the cloud offerings but required by the user [Tomesh et al., 2022]. To enable user-group-specific monitoring and analysis provide suitable abstractions, e.g., by aggregating data or hiding unnecessary information [Beisel et al., 2024].



Resulting Context

The provenance system stores all relevant data produced by the quantum applications, enabling their monitoring and analysis, e.g., to identify errors and optimize the application. Data abstractions facilitate understanding the application and its execution environment, particularly by visualizing crucial data.

Alternating Operator Ansatz (AOA)

Intent

"Approximate the solution of an optimization problem" (Weigold et al. 2021)

Context

To solve a combinatorial optimization problem, a bit string $z=z_1\ldots z_n$ must be found that fulfills a maximum number of m clauses by assigning every binary variable z_i to either 0 or 1. Each of the m clauses involves a subset of these variables. The domain, e.g., all feasible solutions, is either every possible bit string z of length n or a subset of the bit strings. For a bit string z, the value of the objective function $C(z)$ equals the number of clauses that it fulfills:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}}
C(z)=\sum_{j=1}^m C_{j}(z) \text{ where }
C_{\alpha}(z) = \begin{cases} 1,& \text{if }C_{\alpha}\text{ is fulfilled by z}\ 0, & \text{otherwise} \end{cases}

$$

For larger problem instances, a brute force approach (which evaluates the objective function of every solution to find the best solution) is computationally too expensive. Therefore, a heuristic approach that approximates the best solution is also acceptable.

Problem & Forces

Not available

Solution

An Variational Quantum Algorithm (VQA) approach is used to solve the problem: As a first step, the quantum register is initialized with $\newcommand{\state}[1]{{\left| #1 \right>}} \state{s}$ (see solution sketch) which is either a single solution or a superposition of multiple solutions. Preparing the state $\newcommand{\state}[1]{{\left| #1 \right>}} \state{s}$ is assumed to be efficient, i.e., in constant or at most logarithmic depth. Note that this assumption does not hold for all quantum states. After the Initialization, an ansatz is applied. To construct the circuit of the ansatz, a *phase-separating operator* $U(C,\gamma)$ as well as a *mixing operator* $U(B,\beta)$ are used where γ and β are the parameter sets. The phase-separating operator applies a Phase Shift where the phase of a computational basis state $\newcommand{\state}[1]{{\left| #1 \right>}} \state{y}$ is changed according to its value of the objective function $C(y)$:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} U(C,\gamma) \state{y} = f(y)\state{y} $$

E.g., an operator $U(C,\gamma)$ can be defined that applies a shift for every clause fulfilled by a solution $\newcommand{\state}[1]{{\left| #1 \right>}} \state{y}$.

The second operator is the mixing operator which alters the amplitude of the solutions. Thereby, it provides transitions between solutions and especially allows to transition between an arbitrary pair of solutions within the problem domain for some well-chosen parameter β^*. As a result, this operator reflects the domain's structure.

Each iteration on the quantum computer starts with an Initialization of the state $ \newcommand{\state}[1]{{\left| #1 \right>}} \state{s}$ and, then, applies an ansatz circuit which is based on $C(\gamma)$ and $B(\beta)$. The ansatz circuit consists of p alternating unitaries which are drawn from the operators and lead to the following state:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \state{\gamma, \beta} = U(B,\beta_p)U(C,\gamma_p) \ldots U(B,\beta_1)U(C, \gamma_1) \state{s} $$

In the first iteration, the parameter sets γ, β are chosen randomly and $p\in \mathbb{N}$ defines a hyperparameter.
Measuring this state gives z as a single solution which can be evaluated by the objective function C. Sampling this state allows to determine the expectation values for γ and β which is by definition smaller or equal to the maximum of the objective function:

$$ \newcommand{\state}[1]{{\left| #1 \right>}} \left< C\right>{\state{\gamma, \beta}} = \left< \gamma, \beta |C |\gamma, \beta \right> = \bigg \langle \sum x_z \state{z} \bigg | \sum x_z f(z)\state{z} \bigg \rangle \ = \sum |x_z|^2 f(z) \leq \sum |x_z|^2 f(z') = f(z') = C $$

Based on the expectation values, the parameters γ and β can be optimized until the termination condition is satisfied.

Resulting Context

This approach is applicable for NISQ devices and can be adjusted for a particular problem domain. Since NISQ devices are limited by their hardware, only small values for p can be chosen as this hyperparameter determines the width of the circuit. Nevertheless, choosing suitable mixing and phase-separating operators is not trivial for a problem at hand and, currently, an open research question. The convergence of the solution depends on the chosen operators (based on which the ansatz is constructed), the objective function, and the optimization strategy for updating the parameters.

Amplitude Encoding

Also known as: *This encoding has also been referred to as Wavefunction Encoding by (LaRose and Coyle 2020). Every quantum system is described by its wavefunction ψ which also defines the measurement probabilities. By expressing that the wavefunction is used to encode data, it is therefore implied that amplitudes of the quantum system are used to represent data values.*

Intent

Encode data in a compact manner that do not require calculations

Context

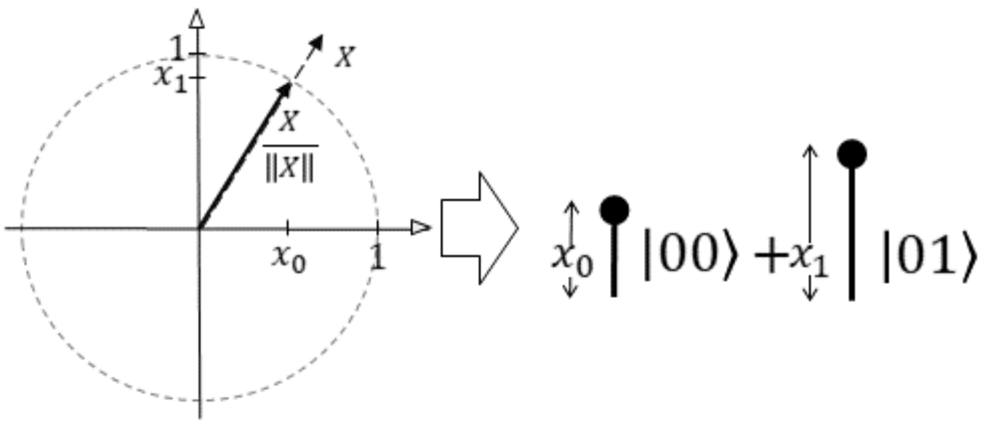
A numerical input data vector $(x_0, \ldots, x_{n-1})^T$ must be encoded for an algorithm.

Problem & Forces

Not available

Solution

Use amplitudes to encode the data. As the squared moduli of the amplitudes of a quantum state must sum up to 1, the input vector needs to be normalized to length 1. This is illustrated in Fig. 5 for a 2-dimensional input vector that contains 2 data points. To associate each amplitude with a component of the input vector, the dimension of the vector must be equal to a power of two because the vector space of an \$n\$ qubit register has dimension \$2^n\$. If this is not the case, the input vector can be padded with additional zeros to increase the dimension of it. Using a suitable state preparation routine (see Known Uses), the input vector is encoded in the amplitudes of the quantum state as follows: \$| \psi \rangle = \sum_{i=0}^{n-1} x_i | i \rangle\$. As the amplitudes depend on the data, the process of encoding the data (but not the encoding itself) is often referred to as arbitrary state preparation.



Pattern sketch, taken from: Weigold, Manuela; Barzen, Johanna; Leymann, Frank; Salm, Marie: Data Encoding Patterns for Quantum Algorithms. In: The Hillside Group (Hrsg): Proceedings of the 27th Conference on Pattern Languages of Programs (PLoP '20).

Resulting Context

A data input vector of length \$l\$ can be represented by \$\lceil log_2(l)\rceil\$ qubits - this is indeed a very compact representation. For an arbitrary state represented by \$n\$ qubits (which represents \$2^n\$ data values), it is known that at least \$2^n\$ parallel operations are needed (Schuld and Petruccione 2018). Current state preparation routines perform slightly better than \$2^n\$ operations (Schuld and Petruccione 2018). However, depending on the data it maystill be possible to realize an encoding in a logarithmic runtime. For example, a Uniform Superposition can be created by applying a Hadamard gate to each of the \$n\$ qubits - which can be done in parallel and thus in a single step. This represents a 2^n-dimensional vector in which all data entries are $\frac{1}{\sqrt{n}}$. Similarly, sparse data vectors can also be prepared more efficiently (Schuld and Petruccione 2018).It must be noted that if the output is also encoded in the amplitude, multiple measurements must be taken toobtain a good estimate of the output result. The number of measurements scales with the number of amplitudes -as n qubits contain 2^n amplitudes, this is costly (Schuld and Petruccione 2018).

Phase Shift

Intent

Distinguish the important aspects of a state in an efficient manner

Context

In an iterative algorithm wherein each iteration the solution shall be improved, the parts of the computational basis improving the solution should be indicated. One possible indication is a phase shift.

Problem & Forces

Not available

Solution

According to (Rieffel and Polak 2014) the subsequent operator S_G^ϕ can be implemented efficiently regarding the number of applied gates:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \sum_{x=0}^{N-1} \alpha_x \state{x} \xrightarrow{S_G^\phi} \sum_{x\in G}e^{i\phi} \alpha_x \state{x} + \sum_{x\notin G} \alpha_x \state{x}$$ This operator marks the qubits which are improving the solution (and, thus, are in the ``good'' subset $G

\subseteq {0, ..., N - 1})$ by a phase shift with a phase ϕ, while leaving the remaining qubits untouched.

Resulting Context

A phase shift of some states in G cannot be detected by measurement, since the amplitudes of the states do not change.
But the phase-shifted states can be used for further computations to increase the probability to measure a "good" solution. Since only the phase and not the measurable amplitude of the states is changed, this operation has no classical equivalent.

Unified Execution

Also known as: –

Intent

How to execute a quantum circuit independently of the heterogeneous quantum cloud offerings and their supported quantum circuit formats?

Context

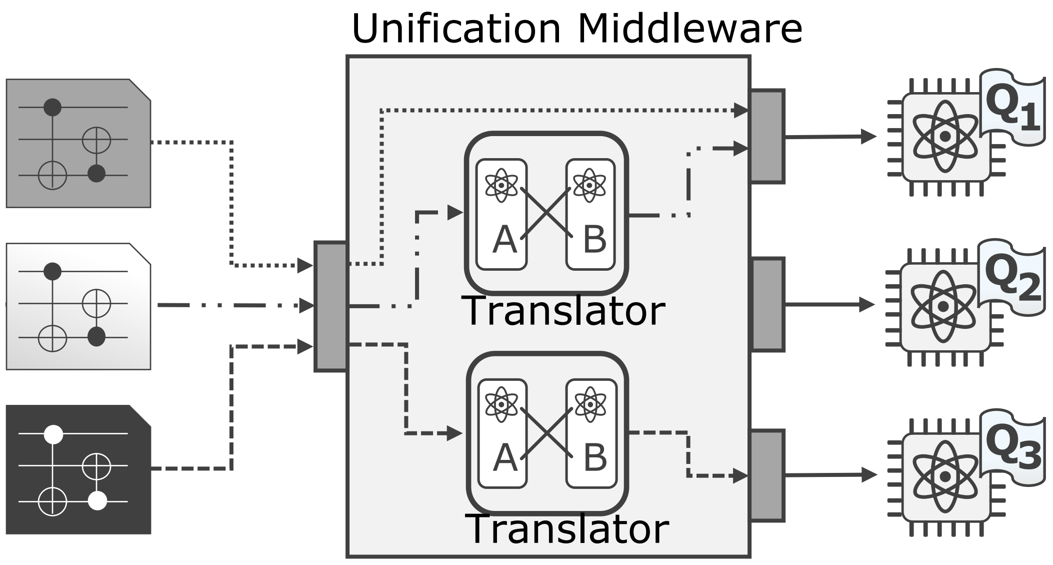
Quantum circuits should be executed using a suitable quantum cloud offering. The circuits might be implemented utilizing different SDKs, such as Qiskit and Braket, quantum programming languages, such as Q#, or quantum assembler, such as OpenQASM [Leymann et al., 2020], [Vietz et al., 2021].

Problem & Forces

When an SDK is used to implement quantum circuits, they can only be executed utilizing a quantum cloud offering supported by the respective SDK. If, on the other hand, a quantum programming language or a quantum assembler is used, the quantum cloud offering for the execution must support this technology. This often leads to a vendor lock-in, which prevents users from flexibly switching to a different quantum cloud offering that provides, e.g., cheaper access.

Solution

Unify the execution of quantum circuits by utilizing a middleware providing a single, unified interface for accessing different quantum cloud offerings. Figure 3 gives an overview of the conceptual structure of the unification middleware. This middleware uses a set of translators that automatically translate the given quantum circuit for the target quantum device if the format of the circuit is not natively supported. Since accessing quantum devices requires users to authenticate themselves, an access token is required for each quantum cloud offering. They can either (i) be provided with each execution request or (ii) the middleware can store the tokens of each user. Alternatively, (iii) the middleware provides access to all quantum devices via a separate pricing model, facilitating the execution for users as they do not require an access token for each quantum cloud offering.



Resulting Context

The quantum circuit can be executed via a single interface independently of the circuit format and quantum cloud offering. Thus, vendor lock-ins can be avoided and circuits can be executed using different quantum cloud offerings without any additional effort. However, the unification middleware does not automatically select a suitable quantum device but only supports the execution on the target device.

Quantum Application Archive

Also known as: –

Intent

How to store, version, and distribute the various heterogeneous artifacts of a hybrid quantum application?

Context

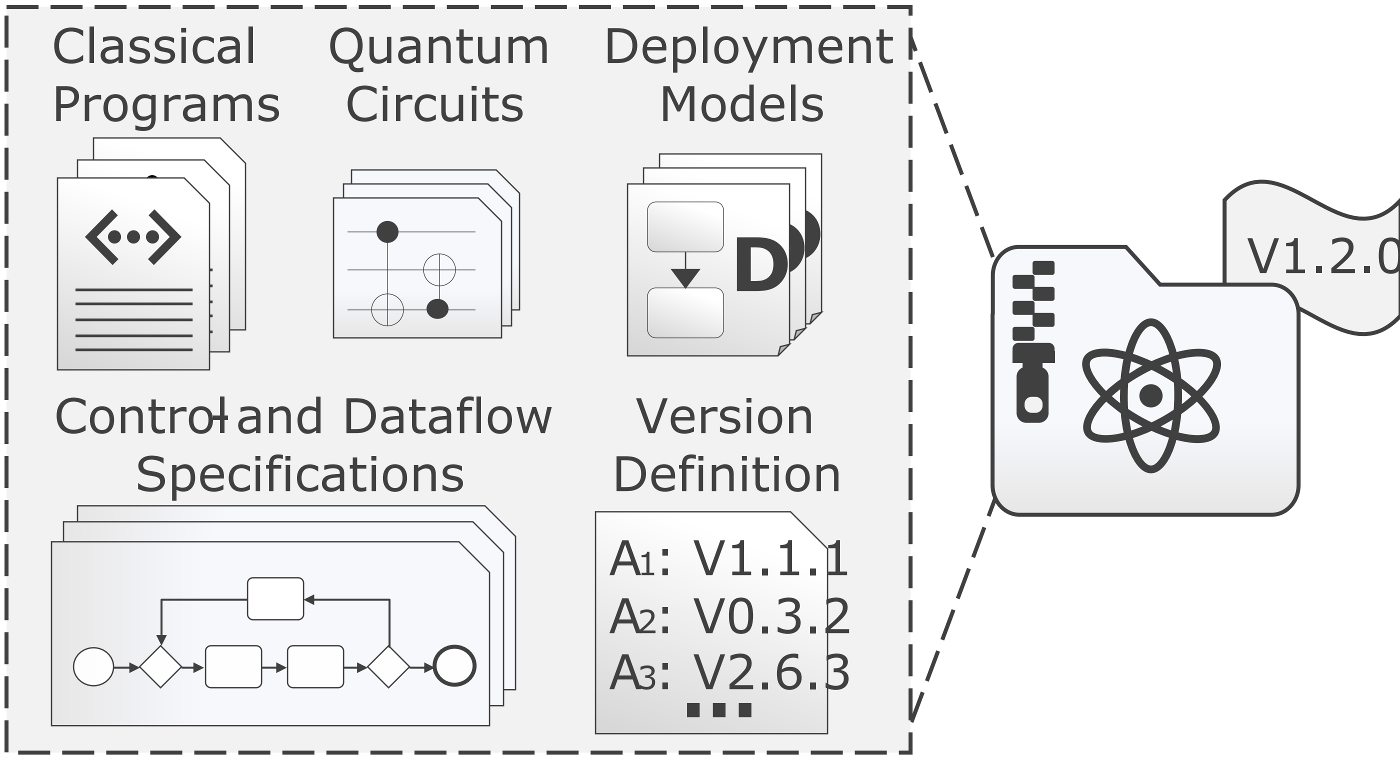
Hybrid quantum applications comprise a wide variety of artifacts, e.g., classical programs, quantum circuits, deployment models, and specifications of the control and data flow [Weder et al., 2022]. The execution of hybrid quantum applications in a target environment requires the availability of these artifacts. For a stable execution in production environments, the application must be versioned and all dependencies must be fixed [Altmanninger et al., 2009].

Problem & Forces

Enter your input for this section here. Software in the quantum computing domain is frequently updated [Vietz et al., 2021], [Weder et al.,2021b]. This often leads to incompatibility when upgrading different parts of the application. Further, identifying all required artifacts and transferring them into a target environment is time-consuming, complex, and error-prone. Distributing and selling hybrid quantum applications, e.g., via a marketplace, typically requires them to be available as a single entity.

Solution

To enable the storage, versioning, and distribution of hybrid quantum applications, package all required artifacts in a self-contained quantum application archive as shown in Figure 2. Thereby, ensure that all artifacts have a fixed version to prevent incompatibilities through future updates. The quantum application archive must comprise all artifacts required to set up the execution environment of the application and to subsequently execute it in this environment.



Resulting Context

By packaging quantum applications utilizing a quantum application archive, all classical and quantum artifacts required for execution can be shared as a single entity. Since the archive does not only contain the executable programs but also comprises all the necessary data for setting up the execution environment, it can be executed independently of the execution environment installed by the user. Due to the holistic versioning of the artifacts of the quantum application archive, incompatibilities are prevented.

Quantum Application Testing

Also known as: –

Intent

How to ensure the correctness of all functionalities of a hybrid quantum application?

Context

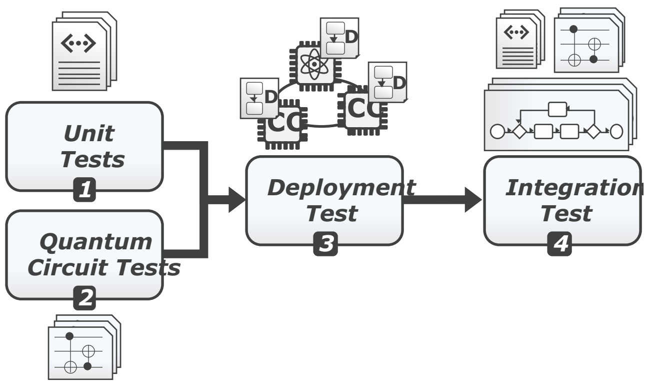
Hybrid quantum applications are realized using a plethora of different artifacts, such as quantum circuits, classical programs, deployment models, and control and data flow specifications Weder et al.,2021b. The correctness of the functionality of all artifacts as well as their interactions must be ensured.

Problem & Forces

Quantum applications comprise heterogeneous programs, e.g., using different programming languages and data formats. The execution of quantum circuits is probabilistic and arbitrary unknown quantum states can not be copied, hence, obtaining information about a qubit without disturbing the corresponding quantum system state is impossible [Ali et al., 2021], [Buzek and Hillery, 1996]. Further, simulating the execution of larger quantum circuits is impossible due to the exponential resources required for simulating additional qubits [Zhou et al., 2020]. The changing characteristics of quantum devices may lead to different results when executing the same quantum circuit at different times, even when using the same quantum devices [Tannu and Qureshi, 2019)].

Solution

Utilize a holistic testing strategy comprising the following steps as depicted in Figure 5: (i) Unit tests for the classical programs. (ii) Specific tests for the quantum circuits. This includes mathematical verification of quantum circuits [Chareton et al., 2021], [Wang et al., 2008], adding and evaluating assertions to ensure certain states [Huang and Martonosi, 2019], [Liu et al., 2020], as well as white and black box tests for quantum circuits [Miranskyy et al., 2020]. (iii) Deployment tests verifying that the application was provisioned as intended (Wurster et al., 2018). (iv) Integration tests validating the interplay of the various software artifacts [Wu et al., 2003].



Resulting Context

By testing all artifacts of a quantum application, as well as their interplay and execution environment, the reliability of the quantum application is significantly increased. Well-tested artifacts of hybrid applications promote their reuse for other applications ([Weder et al., 2022], [Zhao, 2020]. To automate the testing procedure it may be integrated, e.g., into the application's continuous integration and development (CI/CD) pipeline [Romero- Alvarez et al., 2024].

Quantum Hardware Selection

Also known as: –

Intent

How to automatically select a suitable quantum device to execute a given quantum circuit?

Context

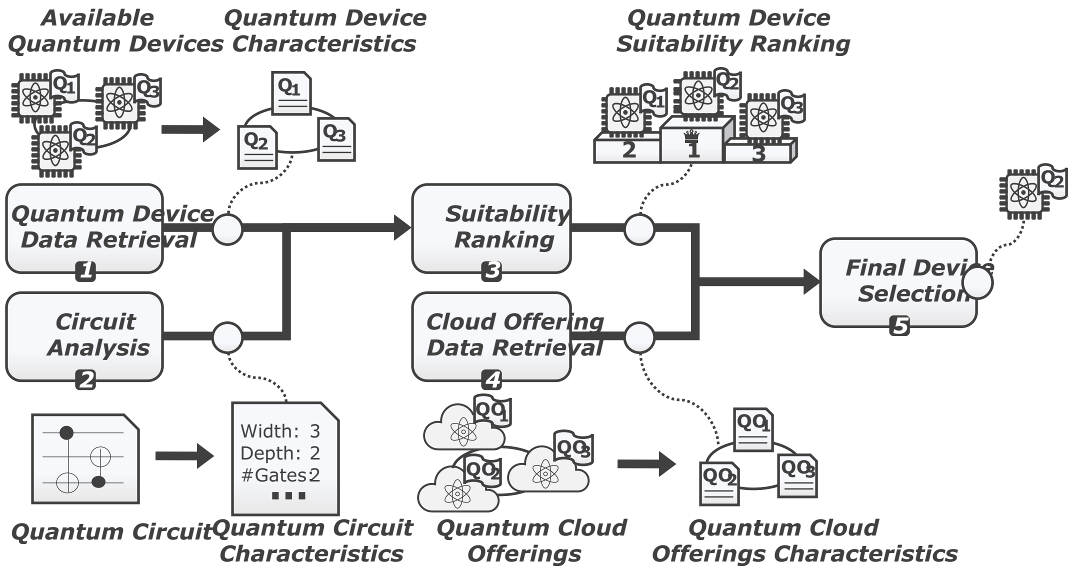
Quantum circuits can either be executed on a quantum device or a classical computer simulating the computation. However, it is impossible to simulate larger quantum circuits using classical hardware [Zhou et al., 2020]. Therefore, a suitable quantum device for the execution must be selected.

Problem & Forces

Quantum devices are provided by different vendors, e.g., IBM, IonQ, and Rigetti [Leymann et al., 2020]. These quantum devices are very heterogeneous and differ in characteristics, such as the number of qubits, their decoherence times, or the supported gate set [Weder et al., 2021a]. Some of the characteristics change over time, e.g., the decoherence times when recalibrating the quantum device [Tannu and Qureshi, 2019)]. However, the successful execution of a given quantum circuit depends on these characteristics [Salm et al., 2020]. Thus, selecting an unsuitable quantum device can lead to error-prone results. Quantum cloud offerings also differ regarding their payment models and access methods, e.g., queue based systems or reservations of exclusive time slots.

Solution

Figure 4 gives an overview of the phases to select suitable quantum devices. In the first step, the characteristics of the quantum devices available to the user are retrieved. This can either be done by periodically accessing an API providing live data about these characteristics, e.g., a provider API or a dedicated provenance system, or by executing benchmarks that approximate device characteristics [Amazon, 2024], [IBM, 2024a], [Weder et al., 2021a]. Then, analyze the given quantum circuit so that in the next phase the suitability of the devices can be ranked based on the characteristics of the circuits and devices [Salm et al., 2020]. Finally, ensure that a quantum device is selected that is available via a suitable cloud offering, e.g., a cloud offering supporting a pay-per-use model.


Available Quantum Devices
Quantum Device Characteristics
Quantum Device Suitability Ranking
Q1
Q2
Q3
1
2
3
Quantum Device Data Retrieval
1
Suitability Ranking
3
Final Device Selection
5
Circuit Analysis
2
Cloud Offering Data Retrieval
4
Width: 3
Depth: 2
#Gates 2
QQ1
QQ2
QQ3
Quantum Circuit
Quantum Circuit Characteristics
Quantum Cloud Offerings
Quantum Cloud Offerings Characteristics

Resulting Context

The quantum circuit can be executed on the selected quantum device. Proper hardware selection might reduce the impact of errors and can optimize other factors, such as the waiting time, depending on the user goal. If the format of the circuit is incompatible with the selected device, it must be translated.

Quantum Classification

Intent

How to train a classifier to assign new data points to one of multiple classes using a quantum device?

Context

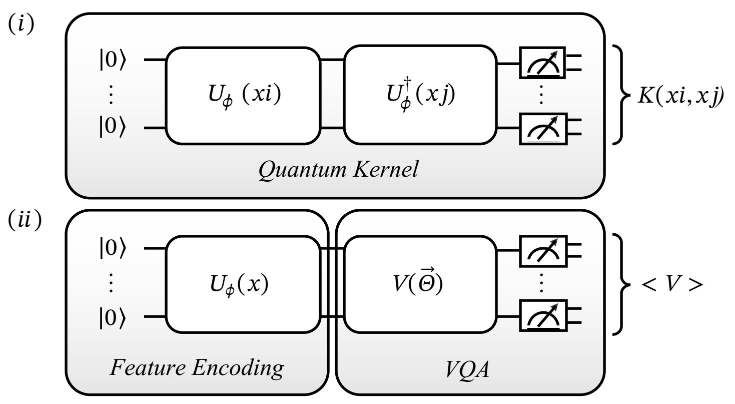
New data points need to be classified into one of several different classes. A labeled set of training data is given.

Problem & Forces

Classifying data is getting increasingly more difficult when the feature space becomes larger [Havlicek et al., 2019]. While quantum computing enables solving this problem by utilizing efficient quantum algorithms, it also leads to additional challenges. For example, high-dimensional data sets can lead to large quantum circuits that may not be successfully executable on today's Noisy Intermediate-Scale Quantum (NISQ) devices [Preskill, 2018]. Additionally, quantum approaches can suffer from exponential cost concentration, which makes models less sensitive to input data, leading to generalization problems [Thanasilp et al., 2024] [Arrasmith et al., 2022].

Solution

Train a classifier using a quantum device to classify new data points precisely. In Figure 3, an overview of two different approaches for training a classifier is depicted. Classifiers can either be trained using (i) a kernel-based method or (ii) a variational method. Generally, the input for training a classifier is an initial set of labeled data $ {(x_i, y_i)}^n_{i=1}$, where x_i are the feature vectors, y_i are the labels, i.e., real numbers, and n is the size of the training set. In the kernel-based approach, a quantum kernel is used to measure the similarities between data points by mapping them into a high-dimensional Hilbert space and computing the inner product of their corresponding quantum states. This quantum kernel is computed for all pairs of training data by applying a unitary $U_ϕ(x_i)$ to encode each data point xi into a quantum state. The adjoint operation $U^{\dagger}_ϕ(x_j)$ is then applied to calculate the overlap between the states corresponding to x_i and x_j. Then, a classical algorithm, e.g., a classical support vector machine Burges, 1998], is used for computing the classifier based on the previously calculated kernel. Alternatively, the variational method optimizes the parameters of a quantum circuit to directly realize the classifier. In this approach, a data point x is first encoded into a quantum state using a unitary $U_ϕ(x)$, which maps the classical data into a quantum state. Once the data are encoded, a parameterized quantum circuit is applied. The circuit produces an output whose expectation value < V > determines the predicted label for a given data point x. The parameters of the circuit are iteratively optimized by a classical optimizer that minimizes a quantum cost function that calculates the differences between the predicted and actual labels from the data set.



Resulting Context

After the training process, the classifier can be used to assign new data points to one of the existing classes. Utilizing a quantum classifier may enable training a more accurate classifier than using a classical classification technique [Kavitha and Kaulgud, 2024]. Quantum classifiers still function under the influence of noise as they are resistant to a small number of misclassifications [Havlicek et al., 2019]. This is particularly important with the noisiness of today's quantum devices. However, mitigation mechanisms must be implemented to address the challenges, such as cost concentration in kernel values or flatness in the optimization landscape [Cerezo et al., 2021] [Thanasilp et al., 2024].

Quantum Neural Network (QNN)

Also known as: *Not available*

Intent

How to learn an unknown unitary operator using a quantum device?

Context

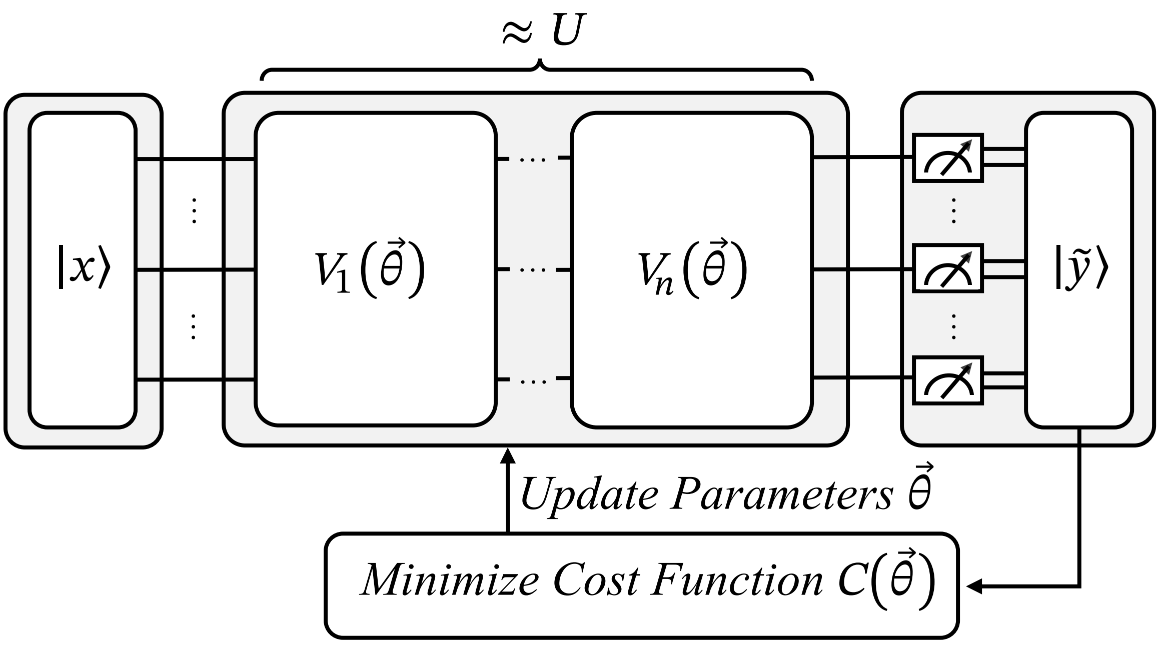
An unknown unitary operator needs to be learned from a training set containing the quantum inputs and the expected quantum outputs.

Problem & Forces

Identifying a unitary operator that is capable of mapping input data to their respective output is getting increasingly more difficult with the complexity and variety of the data. Determining such a mapping requires a lot of input data, and the training procedure requires significant computational power [Achiam et al., 2023]. However, this number can be reduced as outlined by the Quantum No-Free-Lunch Theorem since only obtaining a subset of the training samples as entangled quantum states is already beneficial [Mandl et al., 2023].

Solution

Figure 4 shows the training process of a QNN to learn an unknown unitary operator U: To realize a corresponding quantum circuit, first, the input data are encoded to a quantum state $\newcommand{\state}[1]{{\left| #1 \right>}} \state{x}$. Similarly to classical neural networks, quantum circuits realizing a QNN comprise various parameterized hidden layers $V_i(\vec{θ})$ to approximate U. The parameters $ \vec{θ}$ are iteratively adjusted by an optimizer, which minimizes a cost function until the quantum circuit produces approximately the expected outputs. The cost function uses the expected outputs and similarity measures, such as fidelity, to evaluate how closely the produced outputs $\newcommand{\state}[1]{{\left| #1 \right>}} \state{\tilde{y}}$ match the expected ones $\newcommand{\state}[1]{{\left| #1 \right>}} \state{y}$.



Resulting Context

The result is a set of parameters that configures the QNN to approximate the expected output of the unknown unitary operator. The quality of the approximation depends on the size of the training set, its linear structure, and the degree of entanglement [Mandl et al., 2024]. While entangled data provides benefits for the training of QNNs, a too high level of entanglement can lead to barren plateaus [Thanasilp et al., 2023b].

Hadamard Test

Also known as: *Enter your input for this section here.*

Intent

How to calculate the expectation value of a unitary operator for a given quantum state?

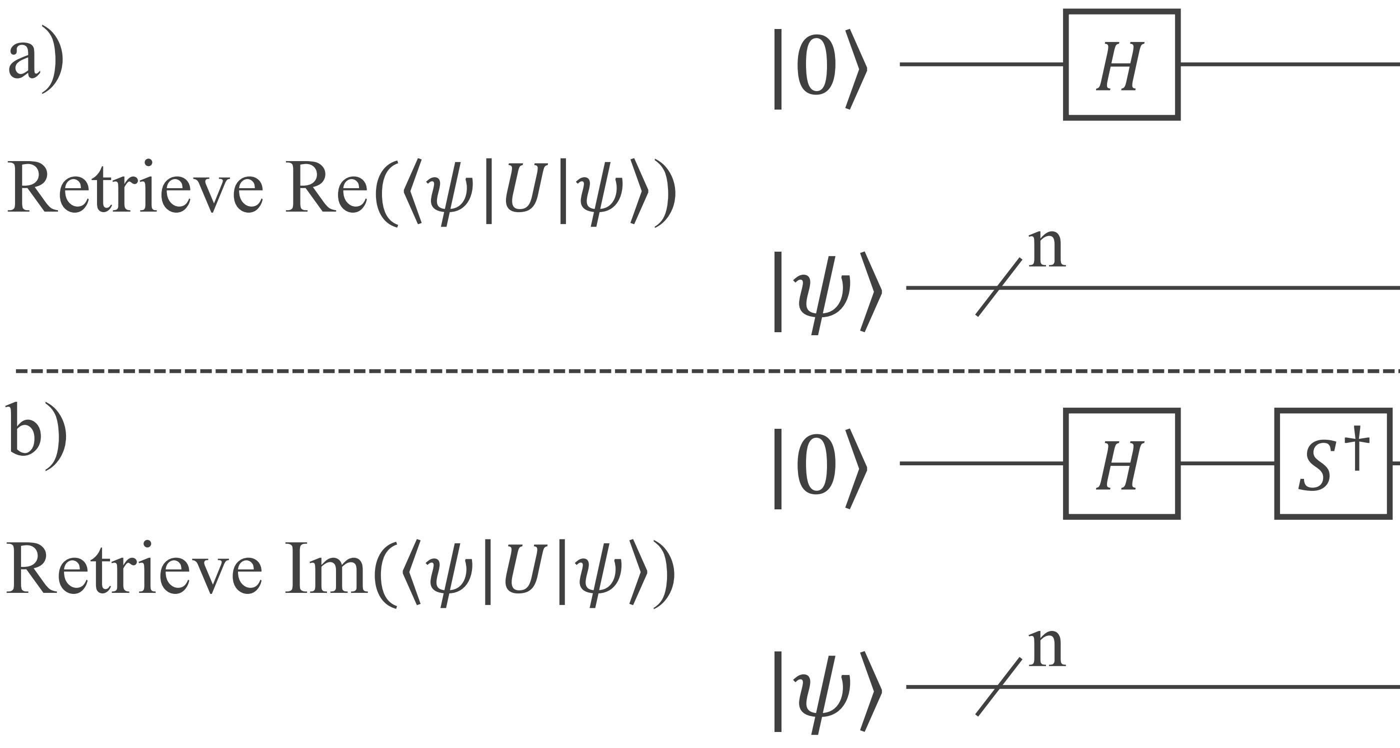
Context

Given a unitary operator U acting on n qubits, and let $\ket{\psi}$ be a n-qubit quantum state. Then, the expectation value $\bra{\psi}U\ket{\psi}$ should be estimated.

Problem & Forces

Determining the expectation value classically is computationally expensive and scales exponentially with the number of qubits [Bravyi and Gosset, 2016]. Quantum devices enable solving this problem more efficiently [Aharonov et al, 2006]. However, using today's quantum devices also leads to additional challenges. For example, large quantum circuits can not be executed successfully, and results are prone to errors [Preskill, 2018].

Solution



The sketch gives an overview of the structure of the quantum circuit, which is required to perform a Hadamard test. The quantum circuit requires one ancilla qubit on which a Hadamard gate is applied to bring it into an equal superposition. Subsequently, depending on whether the real (see a) or imaginary part (see b) of $\bra{\psi}U\ket{\psi}$ should be retrieved, a $S^\dag$ gate is added. Next, the unitary operator U is applied in a controlled manner by using the ancilla qubit as the control qubit. Due to a so-called *phase-kickback*, the information is transferred from the target register to the control qubit when entangling the qubits with the controlled U gate [Ossorio-Castillo et al., 2023]. Finally, another Hadamard gate is applied to the ancilla qubit enabling the retrieval of the expectation value $\bra{\psi} U\ket{\psi}$ by measuring the ancilla qubit.

Resulting Context

The real or imaginary part of the expectation value $\bra{\psi}U\ket{\psi}$ is the output after measuring the ancilla qubit. Depending on the required upper bound of the absolute error, the number of samples must be increased.

SWAP Test

Also known as: *Enter your input for this section here.*

Intent

How to evaluate how similar two given quantum states are to each other?

Context

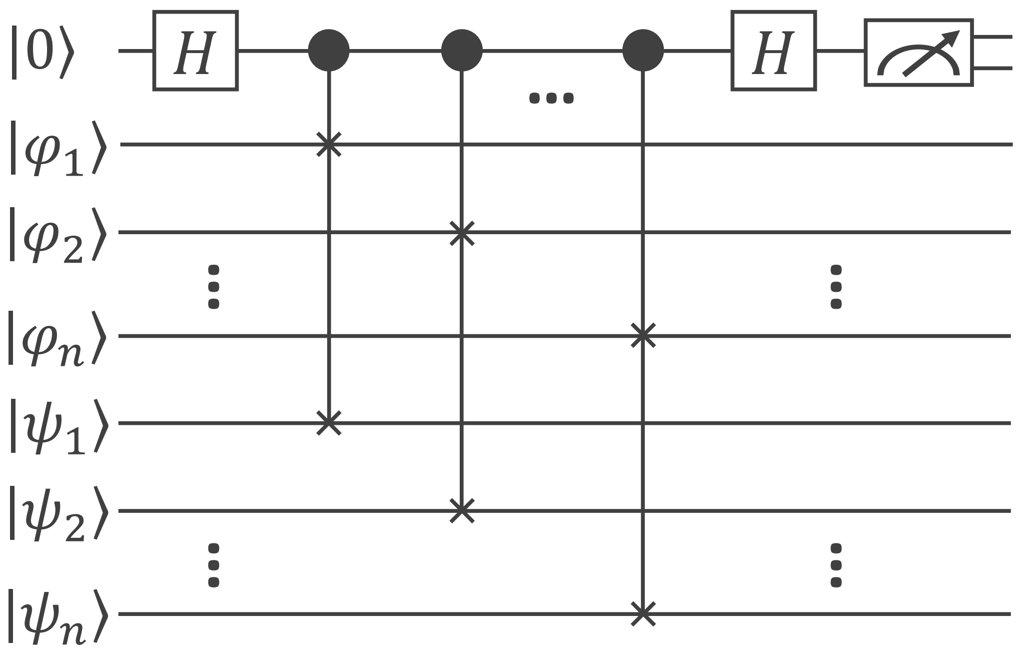
Given two n-qubit quantum states $\ket{\varphi}$ and $\ket{\psi}$. Then, the similarity between these states should be calculated.

Problem & Forces

The similarity of two quantum states may influence the processing of a quantum algorithm. Performing classical measurements is unsuitable for comparing quantum states as the states are destroyed and can not be used for further computations.

Solution

Perform the SWAP test to determine the similarity of the two given quantum states $\ket{\varphi}$ and $\ket{\psi}$.



The structure of the quantum circuit, which is required to perform a SWAP test, is depicted in the sketch. It requires one ancilla qubit on which a Hadamard gate is applied. Next, a sequence of controlled SWAP operators is applied to each qubit of the two states using the ancilla qubit as the control qubit. For example, the first controlled SWAP operation is performed between $\ket{\varphi_1}$ and $\ket{\psi_1}$. Finally, another Hadamard gate is applied to the ancilla qubit, leading to the state: $\frac{1}{2} \ket{0}(\ket{\phi}\ket{\psi}+ \ket{\psi}\ket{\phi}) + \frac{1}{2}\ket{1}(\ket{\phi}\ket{\psi}- \ket{\psi}\ket{\phi})$.

Resulting Context

After the measurement of the ancilla qubit, the outcome determines the similarity between the states $\ket{\phi}$ and $\ket{\psi}$. If the states are identical, the measurement of the ancilla bit results in 0 with probability 1. In contrast, if the states are orthogonal, the measurement results in 0 or 1 with an equal probability of 0.5.

Quantum Fourier Transformation

Also known as: *Enter your input for this section here.*

Intent

How to extract frequencies from a function using a quantum device?

Context

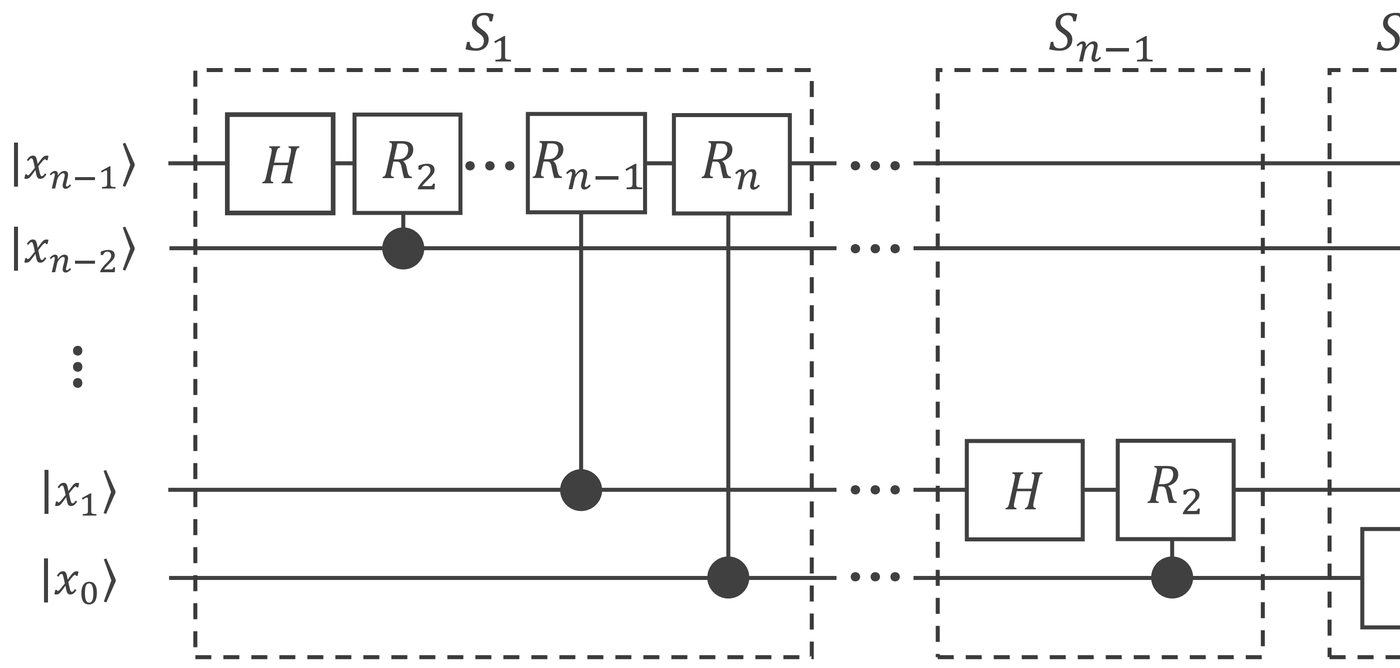
Frequencies need to be extracted from function values, which are given at N distinct points to identify characteristics such as periodicity and distribution of the frequencies.

Problem & Forces

The best known classical implementation of a Fourier transform, the so-called *Fast Fourier Transform (FFT)* is computationally expensive as it requires $O(N \log N)$ operations for a vector that contains N data points of a function [Camps et al., 2020]. While quantum computing enables solving this problem more efficiently, the number of operations executable in sequence on current quantum devices is limited due to high error rates and short decoherence times [Preskill, 2018].

Solution

The QFT extracts frequencies from a n-qubit quantum register $\ket{x} =\ket{x_{n-1},\dots,x_0}$, where $n = ~\log N$. x is interpreted as a decimal number and $x_j \in {0, 1}$ are the binary digits of x. In the QFT, each qubit $ \ket{x_j}$ is transformed as follows: $$ \ket{x_j} \mapsto \frac{1}{\sqrt{2}} \left(\ket{0} + e^{2\pi i x / 2^j} \ket{1} \right) $$



The required stages are illustrated in the sketch. QFT involves n stages, one for each qubit in the quantum register. At stage S_j, operations are performed solely on qubit x_{n-j}, which involves a Hadamard gate, followed by controlled $R_2, \dots, R_{n+1-j}$ gates, except in stage S_n, where only the Hadamard gate is applied. The R_k gate is controlled by qubit $x_{n+1-j-k}$ to realize the phase $e^{2\pi i / 2^k}$. The gate R_k has the form:

$$ R_k = \begin{pmatrix} 1 & 0 \ 0 & e^{2\pi i/2^k} \end{pmatrix}\text{, for } 2 \leq k \leq n $$ This circuit requires $O(n^2) = O((\log N)^2)$ operations, as each stage S_j involves a single Hadamard operation and $n-1+j$ controlled rotations.

Resulting Context

The QFT provides an exponential speedup compared to the FFT. After applying the QFT, the quantum state is transformed into the Fourier basis. If the QFT is applied to a quantum state encoding a periodic function with period p, the resulting state has a high probability of measuring y, where y is a multiple of N/p [Barzen and Leymann, 2022; Dixit and Jian, 2022]. Due to the controlled phase gates, QFT requires a high connectivity between the qubits.

Ad-hoc Hybrid Code Execution

Also known as: –

Intent

How to execute quantum circuits with classical pre- and post-processing steps with no additional deployment or integration requirements?

Context

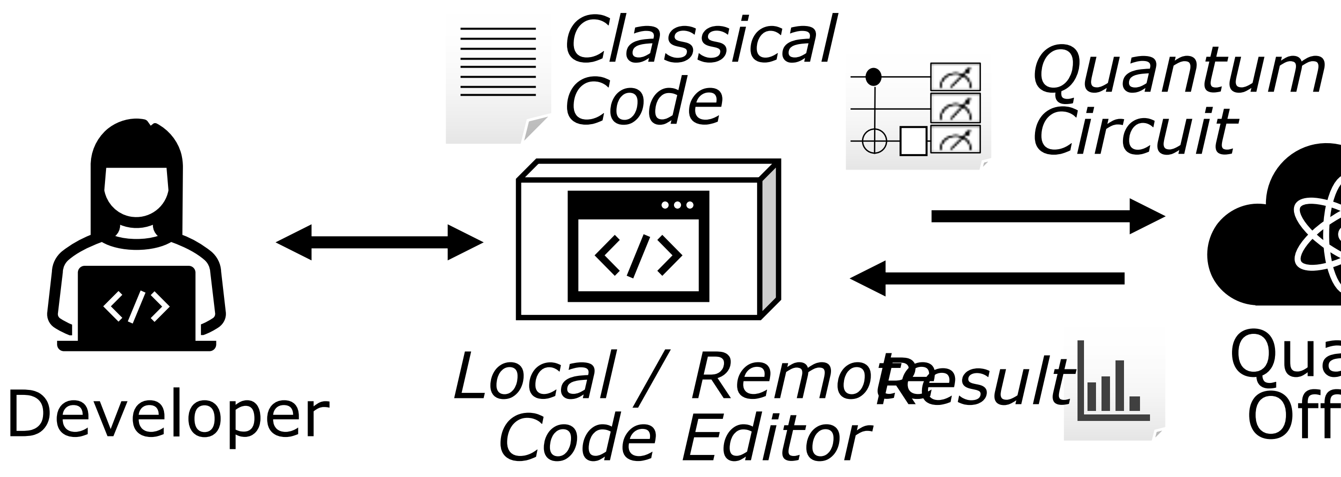
Often it is necessary to execute standalone quantum circuits with their classical pre- and post-processing steps without incurring extra overhead to deploy, run, and manage them in the cloud.

Problem & Forces

Similar to STANDALONE CIRCUIT EXECUTION. However, using this pattern prevents integrating the classical pre- and post-processing steps. Using local or provider-managed development environments reduces management efforts, e.g., execution of quantum and classical code via single commands.

Solution

Use provider-managed code editors or local editors with manually-added token-based authentication. Figure 3 shows the execution of quantum circuits and classical code using AD-HOC HYBRID CODE EXECUTION. The classical pre- and post-processing steps run in a local or remote code editor. The quantum circuits are executed via function calls, often provided by the programming language or quantum SDK, transmitting the quantum circuit to the quantum cloud offering.



Resulting Context

This pattern offers developers more control over the execution of quantum and classical parts, e.g., token management or request construction via SDKs, than the STANDALONE CIRCUIT EXECUTION pattern. However, integration with external applications is often limited, which makes it unsuitable for designing larger applications or running several distinct computations. After a successful ad-hoc execution of quantum circuits and their classical pre- and post-processing steps, all artifacts can be packaged and deployed to a compatible offering or published via application marketplaces to simplify their reuse.

Classical-Quantum Interface

Also known as: –

Intent

How can a quantum algorithm implementation be used by developers without quantum computing knowledge?

Context

Using a quantum algorithm implementation often requires in depth quantum computing knowledge. For example, the Grover search algorithm requires that the user provides a quantum circuit for the missing oracle [Grover 1996]. Other algorithms, like QAOA, require choosing an ansatz, which also requires quantum computing knowledge [Cerezo et al. 2021][Weigold et al. 2021]. However, software developers who want to integrate a quantum algorithm implementation into an application have a deep understanding of the problem domain rather than deep knowledge of quantum computing.

Problem & Forces

To integrate a quantum algorithm implementation into an application, a compatible interface is required. A Hybrid Module already provides an interface enabling its integration into applications, however, using this interface may still require considerable quantum computing knowledge. For example, it may require the problem instance to be provided in the form of a behavior input to the quantum part of an algorithm, or it may have parameters that otherwise influence the quantum part, e.g., by enabling certain error mitigation methods. The effects of the changes, e.g., on resource requirements or runtime, are difficult to estimate without knowledge of quantum computers. Thus, to facilitate the integration of quantum algorithms by problem-domain experts without quantum computing knowledge, such an interface is not sufficient.

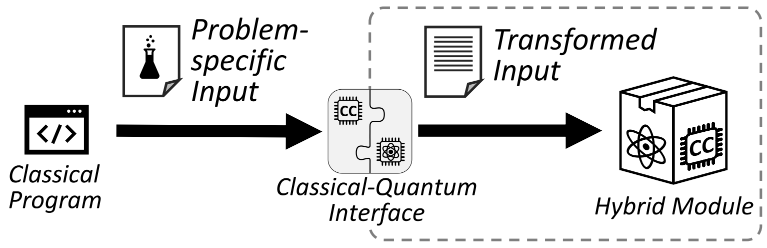
Solution

Use a Classical-Quantum Interface that hides the quantum implementation details. Inputs can be provided to the interface in formats specific to the problem domain.

These problem domain-specific inputs are internally converted into inputs in the formats required by the implementation of the quantum part.

The documentation of interface inputs that affect the quantum part requires special consideration, since understanding their impact on algorithm execution is important information when integrating the quantum algorithm implementation. Thus, the impact of these inputs on the algorithm should be documented in a comprehensible and easily understandable manner by the interface developer. For example, a parameter that increases the accuracy of the result, but also increases the number of gates in the generated circuits, which can result in increased errors with current quantum computers, could be documented as follows:

> “Increasing this parameter can increase the accuracy of the result. However, it also increases the probability of computation errors accumulating, which can negate any improvement in accuracy.“



The sketch shows the interaction of a classical program with a quantum algorithm implemented as a Hybrid Module through a ClassicalL-Quantum Interface. It transforms the problem domain-specific input of the classical program into the inputs required by the quantum algorithm. This interface can also be integrated directly into the Hybrid Module.

Resulting Context

The quantum algorithm implementation can be utilized using a Classical-Quantum Interface. Problem domain experts can make use of this quantum algorithm implementation through the Classical-Quantum Interface created for their domain. The knowledge required to utilize the algorithm implementation is presented in the interface documentation, and the format of input parameters is familiar to problem-domain experts.

Hybrid Module

Also known as: –

Intent

How can the implementation of a quantum algorithm requiring both classical and quantum computations be packaged so that it can be integrated into applications?

Context

Quantum algorithms often require classical computation for pre- and post-processing of the quantum computation results [Leymann and Barzen 2020]. This means that almost all quantum algorithms are hybrid. Thus, any implementation of a quantum algorithm has to contain both the quantum and the classical parts for the algorithm to be functional.

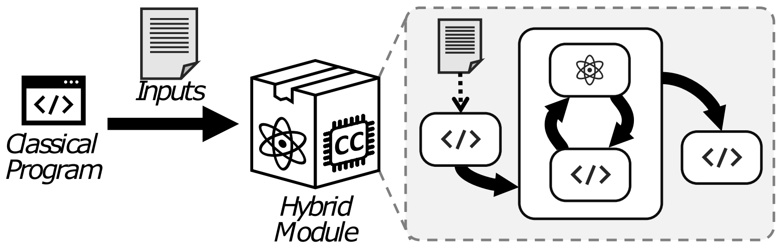
Problem & Forces

Quantum algorithms typically require a classical computer for some parts of their computation. This means that they can have multiple quantum and classical parts. For example, VQAs, such as VQE and QAOA, alternate between quantum and classical computations [Cerezo et al. 2021][Weigold et al. 2021]. Both, the quantum and the classical part, are required for the algorithm to work correctly. This also includes the control flow of the algorithm, which is included in the classical part of the algorithm.

Integrating a quantum algorithm into an application requires the implementation of the entire algorithm. A dedicated interface is required to enable the integration into applications. Deploying the algorithm to a hybrid runtime, which can execute both the quantum and the classical part of the algorithm, even requires both parts to be deployed together.

Solution

Package the entire quantum algorithm, i.e., both the quantum parts and the classical parts, as a Hybrid Module. This module can be composed of smaller modules, e.g., Quantum Modules. It also contains the control flow logic to orchestrate the quantum and classical computation. The Hybrid Module should provide an interface that facilitates its integration into applications. This interface should mainly accept the required problem-specific input values, i.e., the problem that should be processed by the algorithm. Moreover, the interface of a Hybrid Module can also allow behavior inputs to the classical as well as quantum computation, similar to the Quantum Module Templates.



An exemplary sketch of a Hybrid Module is shown in the sketch. It includes the control flow logic and implementations of classical and quantum parts with a loop between quantum and classical computation. The implementation of such a hybrid module can consist of multiple smaller modules, e.g., the three classical and one quantum computation steps shown can each be implemented in a separate module.

Resulting Context

The entire quantum algorithm implementation is packaged as a Hybrid Module. It contains both the quantum and the classical parts, as well as the control flow logic. Hybrid Modules can be used to deploy the algorithm as a standalone service, e.g., in a hybrid runtime environment that can execute both the classical and the quantum part [Riel 2022]. Furthermore, a Hybrid Module can be distributed as a library that implements the quantum algorithm and can be integrated into classical applications. It provides an interface for the application to use. To facilitate the integration of a Hybrid Module by problem-domain experts, a Classical-Quantum Interface can be used as the modules’ interface.

Orchestrated Execution

Also known as: –

Intent

How to ensure the control and data flow for quantum applications comprising one or more quantum circuits with corresponding classical pre- and post-processing steps?

Context

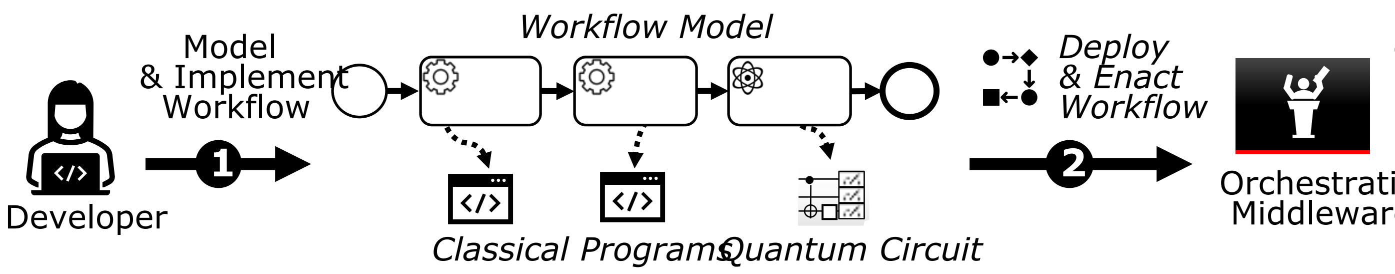
Most quantum algorithms are hybrid, i.e., parts are executed on quantum devices, and others run on classical hardware [Leymann and Barzen, 2020]. Furthermore, quantum applications can involve multiple quantum algorithms and additional classical parts, e.g., interacting with the user or loading data from a database. These parts must be orchestrated, i.e., the control and data flow between them must be ensured.

Problem & Forces

Quantum devices and the corresponding quantum cloud offerings vary strongly in characteristics, such as the number of available qubits, incurred costs for the execution, or queuing times [Tannu and Qureshi, 2019], [Vietz et al., 2021]. The orchestration of parts running in heterogeneous environments can get unmanageable without external orchestration tools, e.g., as there could be long invocation chains, complex data transfers, data format transformations, and interactions with various heterogeneous APIs.

Solution

Utilize a workflow language to model the quantum and classical parts as tasks within a workflow model [Weder et al., 2020b] as shown in Figure 6. The workflow model can then be deployed to a workflow engine, which orchestrates the quantum and classical parts by invoking them in the specified order and ensuring the required data flow (Ellis, 1999). Thereby, the invocation of heterogeneous offerings, as well as features such as data format transformation is provided by the workflow engine [Leymann and Roller, 2000]. While there exist workflow offerings specifically targeting the quantum computing domain, e.g., providing some pre-implemented quantum algorithms, standardized workflow languages and corresponding workflow engines can also be employed to benefit from their maturity and rich feature sets.



Resulting Context

The classical code as well as the quantum circuits required to realize a quantum application are separated from the workflow model defining how they are integrated. This increases modularity and enables the reuse of existing code, decreasing development time and cost. Furthermore, by using workflows, quantum applications can benefit from the reliability, scalability, and robustness of workflow engines [Leymann and Barzen, 2021a]. Finally, also the usage of various heterogeneous quantum and classical cloud offerings with different functionalities is supported. However, the need to model orchestrations to enact them on specialized middleware requires additional expertise and may result in overhead for simple use cases such as circuit design and testing.

Pre-deployed Execution

Also known as: –

Intent

How to execute quantum circuits with classical pre- and post-processing steps that have custom deployment requirements?

Context

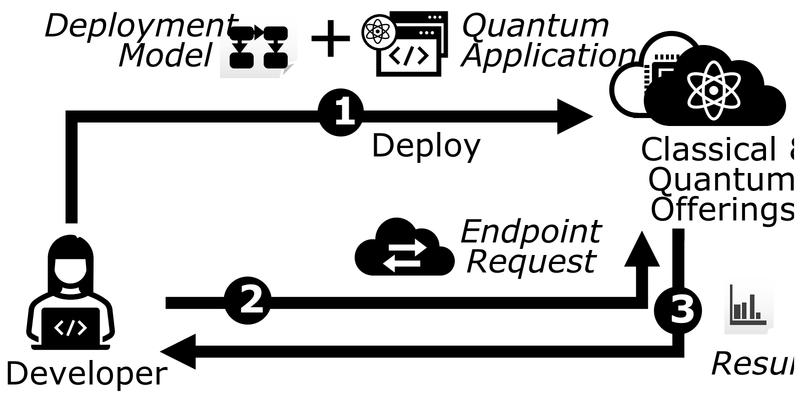
Most quantum algorithms are hybrid, e.g., VQAs contain a hybrid loop with many successive executions of parameterized quantum circuits with optimization steps in-between performed on classical hardware [Cerezo et al., 2021]. Further, quantum devices are accessed via the cloud and to execute a circuit, it is queued in the job-queue of the respective service, which may not support the execution of classical code parts. Thus, a quantum circuit and its classical pre- and post-processing steps may need to be deployed in a specific manner, e.g., together or using specific combinations of cloud service offerings.

Problem & Forces

Quantum service offerings vary significantly feature-wise and often rely on different authentication mechanisms, proprietary formats, and SDKs. Additional technical expertise may be required to successfully execute a provided quantum circuit with its pre- and post-processing steps that can be hosted separately, e.g., due to data processing requirements. In certain cases, it is more beneficial to execute quantum and classical parts of the application in proximity of each other, e.g., to reduce the networking overhead.

Solution

Pre-deploy the quantum circuit with its pre- and post-processing steps either on (i) a single quantum offering that supports execution of quantum and classical parts, or (ii) a specific combination of quantum and cloud offerings that fulfills the given deployment requirements. Figure 4 shows the solution sketch of this pattern: In *Step 1*, the quantum and classical parts are deployed according to deployment preferences. The subsequent execution and fetching of the results shown in *Steps 2&3* can be done independently by developers or client applications.



Variants: One deployment target option in *Step 1* is a Hybrid Execution Environment offering that can speed up the interaction between quantum and classical computations. Hence, they reduce the network overhead and queuing times for many successive quantum circuit executions. Another variant is a Distributed Deployment in which parts of the quantum application are deployed on a combination of different cloud offerings. Implementation of quantum and classical parts, as well as their deployment models, depend on chosen technologies, e.g., implementation of QAOA for Qiskit Runtime would impose more coding and deployment modeling constraints compared to more general deployment scenarios such as packaging the quantum application as one or more containers that can be deployed to a container orchestration engine such as Kubernetes and later executed via client requests.

Resulting Context

The deployment of quantum applications is decoupled from its execution, hence, enabling the invocation by other users or integration with other applications, e.g., a pre-deployed VQA can be subsumed as a part of another application. In the case of hybrid runtimes, the pre-deployed application benefits from provider-managed execution transparency but is locked into the requirements and limitations of the underlying offering. In the Distributed Deployment variant, developers can benefit from combining different services for specific parts of the application.

Mid-Circuit Measurement

Also known as: *Enter your input for this section here.*

Intent

How to extract partial information from a quantum device while a circuit execution is still running?

Context

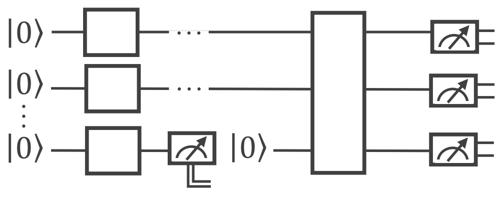
During the execution of a quantum circuit, intermediate results, i.e., information about the state of one or multiple qubits, should be extracted before the circuit execution finally terminates.

Problem & Forces

As quantum devices only provide a limited number of qubits, these qubits must be utilized efficiently, e.g., by reusing ancilla qubits that are no longer required. When measuring a qubit that is entangled with other qubits, the measurement does not only collapse the state of the measured qubit but also irreversibly impacts the quantum states of the entangled qubits.

Solution

Incorporate so-called mid-circuit measurements into the quantum circuit, i.e., measurements before the quantum circuit execution terminates.



As illustrated in the sketch, various quantum operations are performed on different qubits initialized in the $\ket{0}$ state, resulting in an intermediate quantum state. A subset of m qubits is then measured, yielding a measurement outcome $s \in {0,1}^{\otimes m}$, extracting classical information about a part of this intermediate quantum state. Afterwards, the execution of the quantum circuit continues. For example, in the sketch a mid-circuit measurement is performed on the last qubit, measuring a classical 0 and collapsing the state of the qubit into the $\ket{0}$ state.

Resulting Context

Mid-circuit measurements provide information about the intermediate states of the measured qubits. Each measured qubit collapses into the state $\ket{0}$ or $ \ket{1}$, which can, e.g., be used to reset them by applying a controlled X operation after the measurement [Xu et al., 2023]. After resetting the qubits, they can be reused for further computations. By performing mid-circuit measurements on a qubit, the states of qubits that are entangled with the measured qubit are influenced. The states of qubits that are not entangled with the measured qubits are preserved.

Dynamic Circuit

Also known as: *Enter your input for this section here.*

Intent

How to modify a quantum computation during runtime based on intermediate information about a part of the quantum state?

Context

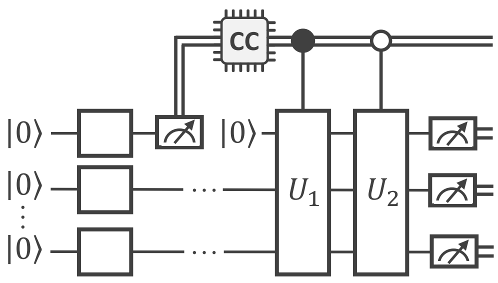
A quantum circuit must be modified based on intermediate information about a part of the quantum state.

Problem & Forces

Measuring a qubit causes its state to collapse and breaks the entanglement between the measured qubit and the entangled qubits. Operations between qubits that are far apart from each other require a lot of intermediate SWAP operations, increasing the depth of the circuit. The low decoherence times of current quantum devices limit the maximum execution time of quantum circuits.

Solution

To modify a quantum computation during runtime based on intermediate information about a part of the quantum state, define a dynamic quantum circuit that utilizes mid-circuit measurements as well as classical processing. Classical processing can either be a feedforward of the measurement results or a more complex computation that utilizes the measurement results to adaptively apply or skip specific quantum operations.



The sketch exemplarily showcases a dynamic circuit using feedforward: First, the quantum circuit performs a sequence of operations. Then, a mid-circuit measurement is performed on the first qubit and the measurement result is used for classical processing. Based on the outcome of the classical processing, it is determined if the gate U_1 or U_2 shall be applied. If the outcome of the classical processing is 1, then U_1 is applied; if it is 0, then U_2 is applied.

Resulting Context

Dynamic circuits enable the development of algorithms and optimization routines that require intermediate information about a part of the quantum state. The classical processing performed after the mid-circuit measurement must be faster than the decoherence time of the quantum device so that the quantum state is not lost before the quantum computation can be modified and completed. Additionally, feedforward of measurement results leads to constant latency for executing conditional operations, while more complex real-time computations introduce additional variable delays depending on their complexity [Gupta et al., 2024].

Prioritized Execution

Also known as: –

Intent

How to execute multiple quantum circuits in succession while keeping the queuing time low?

Context

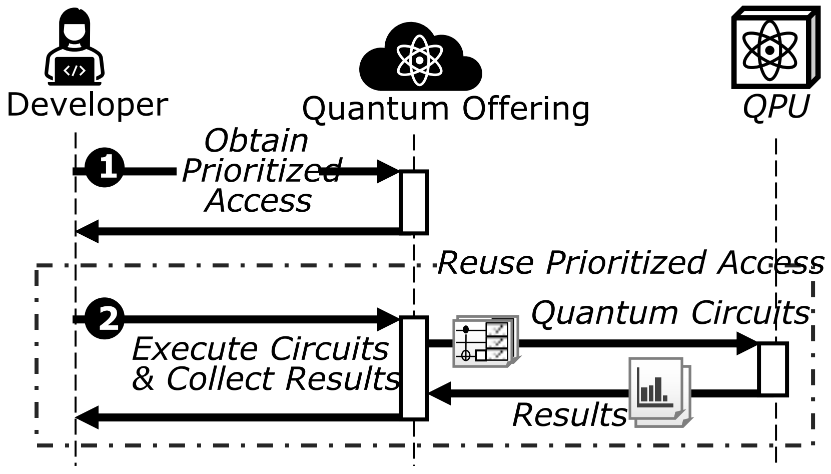
Quantum applications often require executing multiple quantum circuits. This is especially the case when utilizing VQAs on contemporary NISQ devices. Thus, these quantum circuits should be executed efficiently by minimizing the queuing times.

Problem & Forces

Quantum devices are usually accessed via queues, ensuring their fair utilization. Thus, the queuing times sum up when executing multiple quantum circuits independently of each other. One approach is to use batch processing, i.e., combining various quantum circuits into one job, which is then executed at once [Vietz et al., 2021]. However, this approach is not possible when quantum circuits depend on the results of previous executions, e.g., for VQAs.

Solution

Use quantum offerings that enable prioritized access to quantum devices to reduce or completely avoid queuing times, as shown in the solutions sketch in Figure 5. For this, prioritized access to quantum devices is obtained in Step 1. In Step 2, the quantum device can then be reused via this prioritized access, which restricts the number of users to reduce queuing times for multiple circuit executions.



Resulting Context

By applying this pattern, the time spent waiting inside highly occupied job queues is minimized. The choice between the two options is a trade-off between runtime and cost. In the reserved time-slice scenario, runtime is the highest priority since the reservation of a quantum device incurs higher costs than other forms of access. When using sessions for the execution, there can still be competing executions in the job queue from other users.

Quantum Circuit Translator

Also known as: –

Intent

How can a quantum circuit be executed by different quantum computers with different instruction sets?

Context

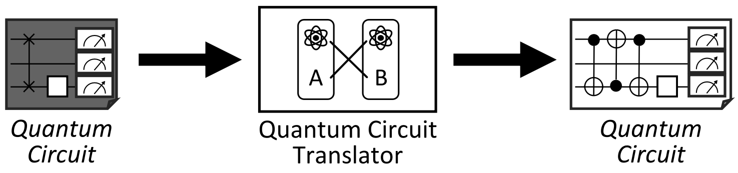
Quantum circuits can be implemented in different programming languages and with different quantum gates. However, quantum computers typically only support specific circuit formats and instruction sets, which hinders interoperability and leads to vendor lock-in [Salm et al. 2020]. Thus, executing a quantum circuit on different quantum computers often requires a translation of the quantum circuit.

Problem & Forces

There are a multitude of quantum programming languages available for implementing quantum algorithms [Vietz et al. 2021]. A quantum circuit may be implemented in a programming language that is incompatible with the targeted quantum computer. The circuit needs to be re-implemented in a compatible quantum programming language and instruction set. However, a manual re-implementation is error-prone, time-consuming, and requires expertise in quantum computing, and, hence, is not feasible for real-world problem sizes. Therefore, an automatic translation, transforming unsupported gates into gates natively supported by the quantum computer, is required.

Solution

Use a translator to convert the quantum circuit into the target language and transpile the circuit to the target instruction set, i.e., replace unsupported gates with equivalent gates from the target instruction set.



The solution sketch shows the application of a Quantum Circuit Translator that translates a quantum circuit between two programming languages and instruction sets. The SWAP gate connecting the outer qubit wires in the left quantum circuit has been decomposed into three C-NOT gates in the right target quantum circuit.

Resulting Context

A Quantum Circuit Translator is able to automatically translate a quantum circuit into a target format, enabling components with different circuit formats and instruction sets to use the same circuit. A Quantum Circuit Translator increases the reusability of Quantum Modules, as it enables their use with different quantum computers. However, the translated circuits do not need to be executed directly, but can instead be used as inputs for a Quantum Module Template. Therefore, a Quantum Circuit Translator enables the composition of quantum algorithms based on modules implemented in different programming languages. Thus, a Quantum Circuit Translator can be used to increase the interoperability of Quantum Modules.

Quantum Module

Also known as: –

Intent

How can the implementation of the quantum part of a quantum algorithm be packaged for reuse independent of concrete input values?

Context

Each quantum algorithm is a hybrid algorithm, i.e., parts of the algorithm require quantum computers and other parts require classical computers for their execution. For the execution of the quantum part, a quantum circuit implementing the required operations is needed. However, quantum circuits are problem-specific and, thus, depend on various inputs, e.g., the problem instance or initial values for parameterized quantum gates, which are then optimized by a classical optimizer. Therefore, the implementation of the quantum part of a quantum algorithm must be input-agnostic in order to be reusable.

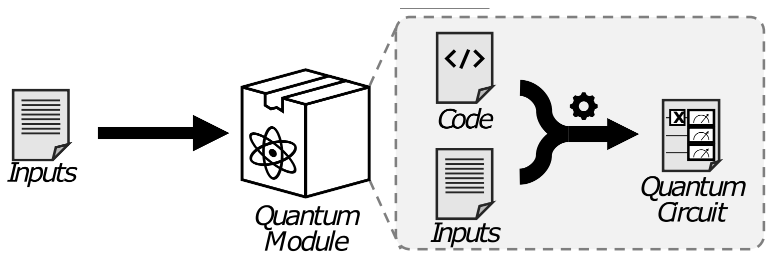
Problem & Forces

Quantum circuits to be processed by a quantum computer must already contain all appropriately encoded input values. A static implementation of a quantum circuit that does not allow the quantum circuit to be changed based on some input values cannot be reused to solve different problems. Thus, a reusable implementation of the quantum part of a quantum algorithm needs to be able to adapt the quantum circuit to different input values. The input values that need to be encoded in a quantum circuit are, first, the problem to be solved, e.g., an implementation of Shor's algorithm [Shor 1997] would require as input the number to be factored into primes, and second, parameters used for optimization or machine learning, e.g., for QAOA [Weigold et al. 2021].

Moreover, implementing the quantum part of a quantum algorithm requires in depth knowledge of quantum computing and the underlying mathematical concepts. Thus, quantum computing experts are required in the development teams. However, other parts of the algorithm that only require classical computation, e.g., classical optimizers, may not require quantum computing knowledge at all and can be implemented by different teams without a quantum computing expert.

Solution

Separate the implementation of the quantum part of the quantum algorithm into one or more quantum modules. These modules contain the code that generates quantum circuits based on input values provided to the module. Quantum modules can also be used to reduce the number of code duplicates by implementing common parts of a quantum circuit as a reusable Quantum Module.



The solution sketch depicts that a quantum module receives input values and uses generative code to construct quantum circuits depending on these input values. This ensures the reusability of the quantum module, as the implementation can create quantum circuits for different problem sizes as well as parameters.

Resulting Context

A quantum algorithm implementation is partitioned into (i) quantum modules containing the implementations of the quantum part, and (ii) additional classical code required for the control flow and other classical computations of the quantum algorithm. The quantum modules are independent of the concrete input values, which increases their reusability for different quantum algorithm implementations.

The separation of code that generates quantum circuits into quantum modules can thus also be reflected in the organizational structure of the development teams. Only the teams working on the quantum modules need quantum computing experts, while other teams mainly need experts in classical software engineering.

Quantum Module Template

Also known as: –

Intent

How can the implementation of the quantum part of a quantum algorithm be packaged for reuse when some of the behavior is determined later?

Context

Some quantum algorithms can be implemented in a reusable manner, but their behavior may be partially modified depending on the problem to which the algorithm is applied. For example, the Grover search algorithm [Grover 1996] contains an unspecified oracle. The information required for defining the concrete behavior of this oracle may not be available until a later point in time. Similar cases are algorithms like QAOA [Weigold et al. 2021], which do not specify a concrete ansatz to use. Thus, implementations of the quantum part of such algorithms, where the unspecified behavior can be integrated later, are required.

Problem & Forces

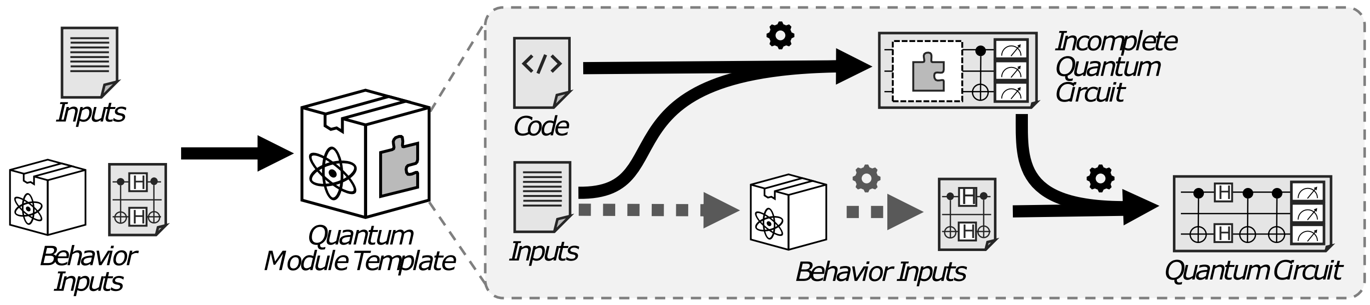
Quantum algorithms may intentionally leave parts of the behavior of the quantum part unspecified until a later point in time. For example, the Grover search [Grover 1996] uses an unspecified placeholder gate, as the specific function that marks the correct values cannot be known before it has been decided what to search for. In the case of QAOA, the choice of a suitable ansatz depends on information that is only available at runtime. However, for the algorithms to be executed, the missing behavior must be integrated before the execution of the quantum circuits on a quantum computer. Note, that similar situations can arise if the development of a quantum algorithm is split between different teams.

Integrating quantum behavior into an existing circuit requires a specification of the requirements an implementation has to fulfill to be integrated and function correctly. This includes the specification of the input qubits available, possible ancilla qubits, on which qubits and in what form the output is expected, and any other requirements or restrictions, e.g., on the creation of entanglement between quantum bits. Some of the restrictions, e.g., the number of available ancilla qubits, may additionally depend on the quantum computer used for execution, as a quantum computer with more qubits can allocate more ancilla qubits if the number of qubits used in the circuit is otherwise constant.

Solution

Implement the generic behavior of the quantum part of a quantum algorithm in a Quantum Module Template. This module accepts inputs, that define the unspecified behavior to be integrated into the final quantum circuit. The behavior can either be specified as a quantum circuit or as a Quantum Module that generates the required quantum circuit. This circuit then gets integrated by the Quantum Module Template into the main quantum circuit that represents the generic behavior.

To ensure that the behavior input, in form of a quantum circuit, can be integrated to correctly perform the operations it contains, the Quantum Module Template must include specifications in the documentation that can be used to build a compatible quantum circuit, as outlined in the pattern forces. This specification is mainly a contract that needs to be fulfilled by the quantum circuit serving as input for the template. Similar contracts, e.g., plugin contracts Marquardt 1999, are also used in classical software engineering.



The sketch shows the essential building blocks of a Quantum Module Template. The template requires two kinds of inputs: (i) the input values representing the problem to be solved as well as parameters affecting the circuit generation, as used in theQuantum Module, and (ii) behavior inputs partially specifying the behavior of the algorithm, provided in the form of a quantum circuit or a Quantum Module. Much like the Quantum Module, the Quantum Module Template uses the input values to generate a quantum circuit, which is still incomplete as it does not include the behavior from the behavior inputs yet. If the behavior inputs are provided in the form of a quantum module, this module is used to generate a quantum circuit from the inputs. Finally, the quantum circuit is integrated into the incomplete main circuit. However, implementations of the template are not limited to the exemplary steps shown here.

Resulting Context

The generic behavior of the quantum part of a quantum algorithm is implemented as Quantum Module Template that requires behavior inputs to generate an executable complete quantum circuit. The behavior inputs specify the parts of the algorithm's behavior that cannot be known in advance. Thereby, their influence on the resulting quantum circuit can be significantly higher than with a Quantum Module. The behavior inputs must be compatible with the required input definitions of the template.

The integration of the behavior inputs can be done at design time if the behavior is provided as Quantum Module, since the Quantum Module generates the circuit based on the input values. Templates can be nested inside other templates to compose quantum circuits from Quantum Modules implementing higher level circuit functions. This facilitates the replacement of a part of a quantum circuit if that part should be generated by a new Quantum Module implementing an improved algorithm, e.g., a more efficient state preparation.

Readout Error Mitigation

Also known as: *Enter your input for this section here.*

Intent

How to reduce the impact of erroneous measurements such that the measured result is closer to the intended quantum state?

Context

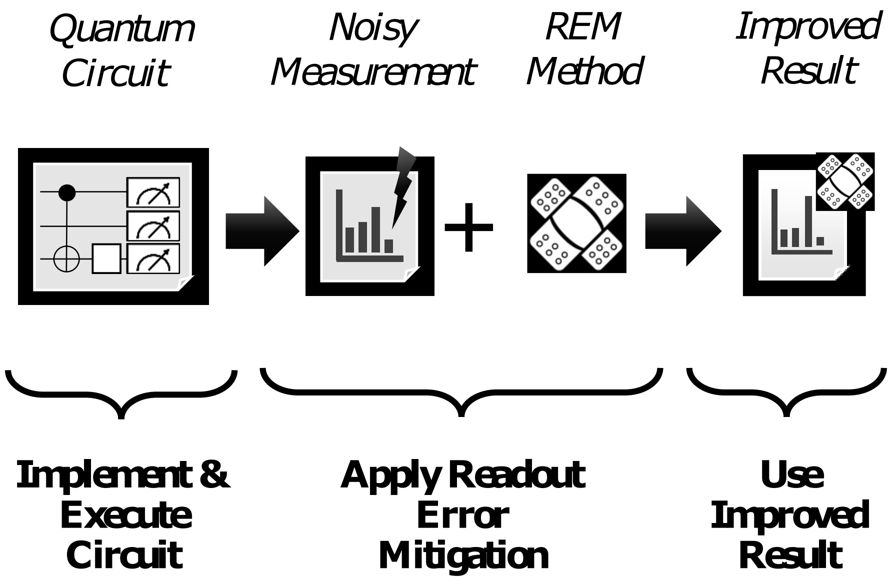
A NISQ-compatible quantum algorithm, e.g., QAOA or VQE, needs to be run on a quantum device. The device’s decoherence times are short and the measurement operations are error-prone. Hence, the measured probability distribution is inaccurate, even when the measured quantum state is accurate. Thus, the negative impact of readout errors needs to be mitigated to obtain a precise measurement result.

Problem & Forces

The measurement times of quantum computers in the NISQ era are significant in comparison to their decoherence times. Therefore, the measurements are highly error-prone and often are among the main error sources. Due to the limited capabilities of current NISQ devices, a minimal number of additional qubits and quantum gates shall be used for the mitigation of readout errors. Further, a quantum device’s measurement error rates change over time, thus the Readout Error Mitigation (REM) needs to be adaptive.

Solution

Mitigate the impact of readout errors by applying a REM method. The mitigation method is performed after the circuit execution and adjusts the measured probability distribution. The resulting mitigated probability distribution is a more accurate representation of the intended quantum state. A solution sketch for the application of REM is shown in figure below. First, the quantum circuit is implemented and executed. Then the resulting probability distribution is improved based on measurement characteristics collected for the quantum device. These characteristics are typically obtained by separately running so-called calibration circuits. Alternatively, adapted instances of the implemented circuit can be run to obtain additional information about the measurement properties.



Resulting Context

REM can reduce the impact of errors caused by measurement operations. The resulting, more precise probability distributions make NISQ devices more suitable for real-world use cases. However, additional classical processing is necessary, which can significantly increase the runtime and classical resource requirements, as not all mitigation methods scale well with the number of qubits. Generally, data provenance can be employed to increase the efficiency of frequently occurring REM tasks, e.g., when executing a VQA.

Quantum Approximate Optimization Algorithm (QAOA)

Intent

Approximate the solution of an optimization problem Weigold et al. 2021

Context

To solve a combinatorial optimization problem, a bit string $z=z_1\ldots z_n$ must be found which assigns each binary variable z_i to either 0 or 1 and fulfills a maximum number of m clauses. Hereby, each clause involves a subset of the variables. In contrast to the potentially constrained domain of solutions in Alternating Operator Ansatz (AOA), every bit string $z=z_1\ldots z_n$ of length n is a solution to the problem.

Problem & Forces

Not available

Solution

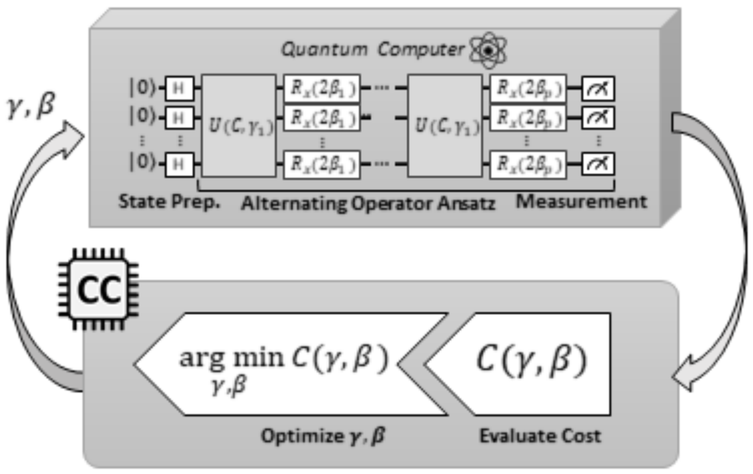
The overall structure of the *Quantum Approximate Optimization Algorithm (QAOA)* (Farhi and Goldstone 2014) approach is depicted in the solution sketch.

First, a Uniform Superposition realizing all possible solutions in Basis Encoding is prepared. E.g., the solution with all binary values assigned to 0 is represented by $ \newcommand{\state}[1]{{\left| #1 \right>}} \state{0 \ldots 0 0}$ and is contained in the superposition.

Then, an ansatz circuit is applied that is constructed based on the two operators $U(C,\gamma)$ and $U(B,\beta)$:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} U(C,\gamma) = e^{i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}; \ U(B,\beta) = e^{-i\beta B} = \prod_{j=1}^{n}e^{-i\beta \sigma_x^{j}} $$

The first operator is a phase shift $e^{-i\gamma}$ on every computational basis state for every clause that is fulfilled. However, this marks but does not change the amplitude of computational basis states (which each represent a solution), thus, the second operator $U(B,\gamma)$ is required. $U(B,\gamma)$ defines a rotation around the X-axis for every qubit whereby the angle for the rotation depends on γ. Based on the structure defined in Alternating Operator Ansatz (AOA), a trial state $\newcommand{\state}[1]{{\left| #1 \right>}} \state{\gamma,\beta}$ is prepared. Measuring this state results in a single bitstring, i.e., a solution that can be evaluated by the objective function. The parameters β and γ which are initialized randomly for the first iteration can then be adjusted. This iterative process continues until the termination condition is satisfied.



Pattern sketch, taken from: Weigold, Manuela; Barzen, Johanna; Leymann, Frank; and Vietz, Daniel: Patterns For Hybrid Quantum Algorithms. In: Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2021).

Resulting Context

The depth of the overall circuit is at most $mp + p$ (Farhi and Goldstone 2014) which is rather shallow. This is one reason why this algorithm is considered a promising candidate for NISQ devices. From a theoretical point of view, it can be noticed that the algorithm approximates the best solution if suitable small values for the parameters γ, β are chosen with $p\rightarrow \infty$. Nevertheless, note that the performance of the algorithm also depends on the objective function and the optimization strategy.

Standalone Circuit Execution

Also known as: –

Intent

How to execute standalone quantum circuits that impose no deployment or integration requirements?

Context

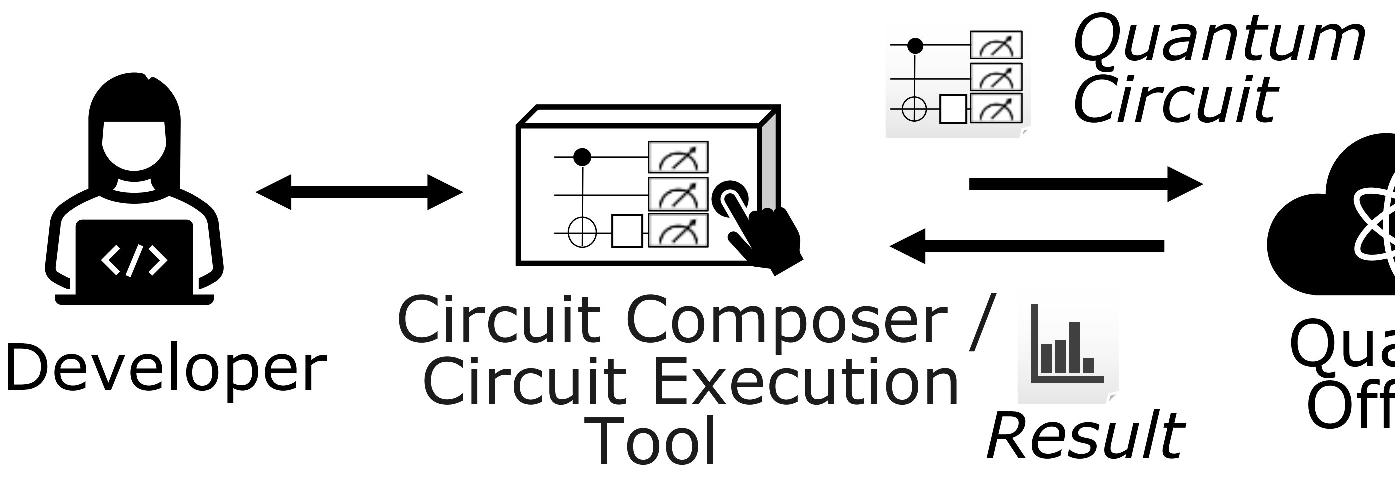
In some scenarios, only standalone quantum circuits need to be executed, such as circuit design and testing or education. Thereby, it is beneficial to execute quantum circuits with minimal effort and required knowledge about different quantum offerings as well as deployment or integration technologies.

Problem & Forces

Quantum offerings vary in features and capabilities. Use of advanced offerings for simple use cases may incur unnecessary management overhead and even block the task due to lacking technical expertise, e.g., deployment automation and integration technologies. In contrast, certain quantum offerings reduce the amount of required management efforts, e.g., by generating execution requests automatically.

Solution

Execute quantum circuits via quantum offerings that do not require implementing custom deployment logic or integration with other services. Figure 2 shows the solution sketch in which distinct quantum circuits are executed using offerings capable of creating and executing circuits. This includes dedicated graphical circuit composers or text-based tools transmitting the quantum circuits to the offering. Typically, providers are responsible for the majority of the management efforts, e.g., deployment of circuits and authentication to the underlying quantum offering.



Resulting Context

Quantum offerings supporting this pattern provide a simple way for executing quantum circuits. However, classical pre- and post-processing steps can not be defined using these offerings. Thus, this style of execution is not suitable for running larger quantum applications and integration with external applications is either very limited or not supported at all. Additionally, running computations with this pattern cannot take advantage of prioritized executions.

Wire Cut

Also known as: *Enter your input for this section here.*

Intent

How to interrupt a wire in a quantum circuit classically such that no quantum information is transmitted while preserving the computation's result?

Context

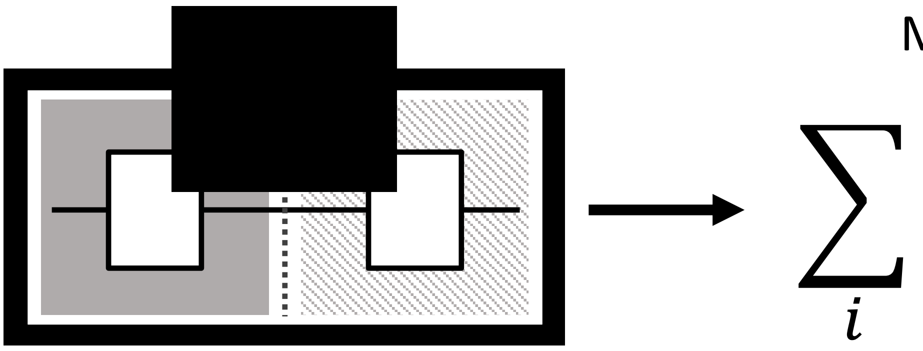
In a quantum circuit, a wire transfers a non-trivial quantum state between two gates, that is, a state that exhibits superposition and has the potential of entanglement with other qubits. The precise state conveyed by the wire is unknown. Furthermore, the quantum state experiences decay over time. Moreover, the interest of the experiment lies in the expectation value over multiple shots rather than the outcome of a single-shot execution.

Problem & Forces

Given the non-trivial nature of the state, it cannot be measured and reinitialized based on the measurement outcome, as the qubit's full state is not revealed through the measurement process. Instead, the measurement collapses the qubit probabilistically to a state associated with the measurement basis, destroying its superposition and entanglement. Furthermore, since the state of the wire is unknown, it cannot be restored independently of the previous measurement either.

Solution

Apply a wire cut to decompose a circuit's wire into a weighted sum of subcircuits [Peng et al., 2020]. As shown in the solution sketch, the subcircuits interrupt the wire with different measurements described by observables O_{i} and subsequent qubit initializations of the states $\ket{\psi_{i}}$. The sum of the expectation values of these subcircuits, each weighted by real coefficient a_i, must replicate the expectation value of the original circuit. To cut the wire, run the subcircuits and combine their results based on the coefficients a_{i} and the measurement outcomes.



Resulting Context

Applying a WIRE CUT interrupts the wire classically without disturbing the computation's expectation value. The behavior of superposition and entanglement is simulated by replacing the wire with a linear combination of subcircuits that perform a measurement and then reinitialize the qubit. However, the trade-off involves executing multiple subcircuits, which raises the computational overhead in terms of shots needed to maintain the same statistical accuracy in the computed expectation value as the original circuit. To reduce this computational overhead, it is advantageous to incorporate classical communication in the cut, enabling each state preparation to depend on its immediate prior measurement outcome [Brenner et al., 2023, Pednault, 2023, Harada et al., 2023]. This approach also lowers the cost of cutting multiple wires together compared to cutting each wire individually. Furthermore, leveraging multiple devices for parallel execution of subcircuits decreases the total computation runtime [Bravyi et al, 2022]. Additionally, employing maximum-likelihood fragment tomography can enhance subcircuit results and thereby improve the overall output quality [Perlin et al, 2021]. However, a wire cut only reproduces the result regarding the expectation value, and therefore, it is not suited for single-shot experiments [Brenner et al., 2023].

Gate Cut

Also known as: *Enter your input for this section here.*

Intent

How to partition a multi-qubit gate into independent gates while preserving the computation's result?

Context

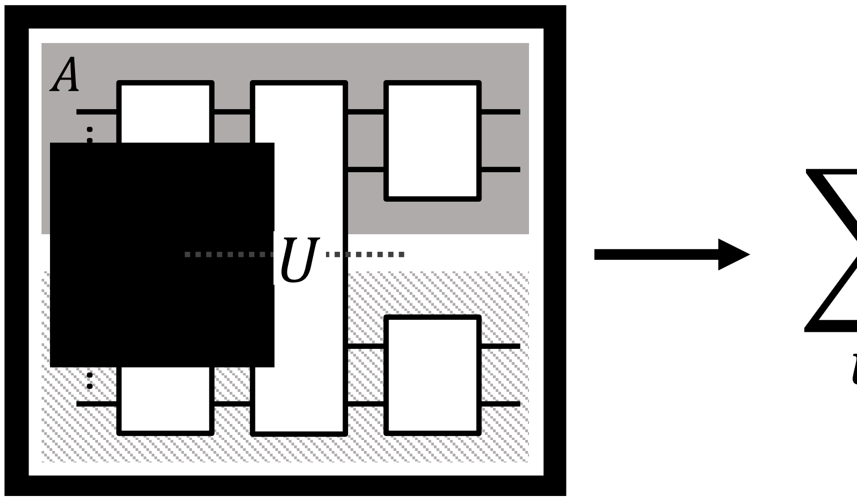
A quantum circuit contains a multi-qubit gate operating between two partitions of qubits and cannot be expressed as a single product of local gates, i.e., gates that operate exclusively on qubits within a single partition. This gate can create entanglement between the partitions, and its execution requires direct physical connections between the involved qubits. The focus is on the expectation value across multiple shots, not single-shot experiments.

Problem & Forces

Limited connectivity between qubits of the multi-qubit gate requires compensating with SWAP operations that exchange the states of two qubits to establish required connections, thereby introducing additional errors [Leymann et al., 2020]. The complete absence of connectivity between the qubit partitions, e.g., multiple devices without quantum communication, renders the execution of gates between partitions impossible.

Solution

Apply a gate cut to decompose a circuit's multi-qubit gate into a weighted sum of subcircuits Mitarai et al., 2021. As shown in the solution sketch, in each subcircuit, the original multi-qubit gate U is replaced by local operations F_{i}^{A} and F_{i}^{B}, which act on two disjoint qubit partitions labeled A and B. The sum of the expectation values of these subcircuits, each weighted by real coefficient a_i, must replicate the expectation value of the original circuit with gate U. The local operations F_{i}^{A} and F_{i}^{B} can be either unitary transformations or projective measurements. Execute these subcircuits and combine their results using the real coefficients a_{i}.



Resulting Context

The GATE CUT enables implementing a multi-qubit gate as a linear combination of local operations independent of the qubit connectivity. This eliminates the need for additional SWAP gates when applied on a single device and allows a multi-qubit gate across two quantum devices without requiring quantum communication. However, replacing multi-qubit gates with local gates and projections through executing multiple subcircuits increases the computational overhead in terms of required shots to achieve the same statistical accuracy in the expectation value as the original circuit. To reduce computational overhead when cutting multiple gates, consider incorporating classical communication between the partitions [Piveteau et al, 2024]. Executing subcircuits in parallel on multiple devices reduces the overall computation runtime [Bravyi et al, 2022]. Moreover, this approach reproduces only the expectation value, making it unsuitable for single-shot experiments [Piveteau et al, 2024].

Variational Quantum Eigensolver (VQE)

Also known as: *An alias for this pattern is Quantum Variational Eigensolver (QVE) (Mitarai et al. 2018).*

Intent

"Approximate the lowest eigenvalue of a matrix" (Weigold et al. 2021)

Context

A hermitian matrix H is given, for which the lowest eigenvalue has to be determined. Since the current NISQ devices cannot provide the resources needed to make use of the quantum phase estimation algorithm, an alternative approach must be used.

Problem & Forces

Not available

Solution

As a first step, H is re-written as a weighted sum of Pauli strings:

$$ H=\sum_{\alpha} h_\alpha P_\alpha $$

Using the iterative structure of the Variational Quantum Algorithm (VQA), a suitable ansatz is chosen for the preparation of trial states. Popular choices include the unitary coupled cluster ansatz (Taube and Bartlett 2006) or an ansatz inspired by hardware (Cerezo et al. 2020). The objective function is then defined as follows:

$$C(\theta) = \left<\psi(\theta)|H|\psi(\theta) \right> = \sum_{\alpha} h_\alpha \left< \psi(\theta)|P_\alpha|\psi(\theta) \right>$$

i.e., the solution is evaluated by the sum of the expectation values of the Pauli string. The variational principle guarantees that the sum of expectation values is greater than or equal to the smallest eigenvalue which must be approximated. If the termination condition has not been fulfilled by $C(\theta)$, the parameters are further optimized and updated as described in the Variational Quantum Algorithm (VQA).

Resulting Context

Since a Variational Quantum Algorithm (VQA) approach is used, the convergence of the overall algorithm depends on the objective function, the ansatz, and the chosen optimization strategy. Based on the outcome of this algorithm (an approximation of the lowest eigenvalue), a follow-up algorithm for finding other eigenvalues of H can be applied (see known uses). If all eigenvalues are known, a principal component analysis can be done to perform a dimension reduction.

Warm Start

Intent

"Fine-tune an optimization algorithm by warm starting it" (Weigold et al. 2021)

Context

The best solution for an optimization problem must be found or approximated. For classical methods, the *Unique Game Conjecture* (UGC) states that there is a theoretical upper bound for the approximation ratio which can not be further improved. This implies that classical methods can only approximate up to this bound, i.e., up to a certain extent. However, since the UGC is not true when entanglement is used, quantum algorithms have the potential to surpass these bounds, i.e., they can approximate better solutions than classical algorithms.

Problem & Forces

Not available

Solution

Start by using a classical approach to approximate the best solution. The classical approximation result can often be found as a solution for a related problem, e.g., by weakening or eliminating constraints of the problem.

Resulting Context

Using the warm starting approach, a proceeding optimization is initialized with the classical approximation result which should be nearer to the optimum than a random starting point. The warm starting procedure should also be taken into account for the overall runtime complexity.

Variational Quantum Algorithm (VQA)

Intent

"Optimize the parameters of a quantum circuit on a classical computer" (Weigold et al. 2021)

Context

The best solution for a problem must be found or approximated across all possible solutions. An individual solution can be evaluated by an objective function C that is also given.
By definition, this function is *faithful*, i.e, its minimum value indicates the best solution (Cerezo et al. 2020). Preferably, C is also *operationally meaningful* which means that solutions can be compared: smaller values of C also indicate better solutions. Since the number of possible solutions increases exponentially with the size of the problems, it is too expensive in terms of computations to evaluate all possible solutions.

Problem & Forces

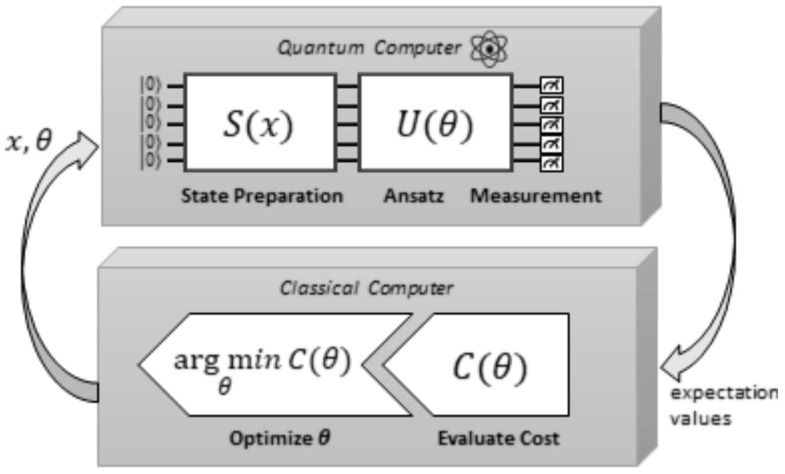
Not available

Solution

A hybrid setup is used to evaluate and optimize solutions. On the quantum computer, an initial state is created that may also encode or be varied according to a set of input data x (refer to the quantum circuit in the upper part of the solution sketch). On this state, an ansatz $U(\theta)$ is applied which is a circuit that depends on a set of parameters θ. This results in the $\newcommand{\state}[1]{{\left| #1 \right>}} \state{\psi_{out}(x,\theta)}$ state. A canonical example of an ansatz is to apply multiple one-qubit operations defining a rotation in the Bloch Sphere around a rotation angle that depends on θ. However, plenty of other parameterized circuits have been proposed as an ansatz. Then, based on the expectation values of the output state $\newcommand{\state}[1]{{\left| #1 \right>}} \state{\psi_{out}(x,\theta)}$, the objective function C for the parameter values $ \theta$ is calculated by the classical computer:

$$ \newcommand{\state}[1]{{\left| #1 \right>}} C(\theta)=\sum_i f_i(\left *{\state{\psi)* $ $ }(x,\theta)}

where O_i is the observable and $\newcommand{\state}[1]{{\left| #1 \right>}} \left *{\state{\psi*$ the expectation value of the i-th measurement to which the function f assigns an overall cost. If $C(\theta)$ is sufficiently low, i.e., the termination criteria are fulfilled, the algorithm ends. Otherwise, the parameter set θ is optimized further for the next iteration. }(x,\theta)}



Pattern sketch, taken from: Weigold, Manuela; Barzen, Johanna; Leymann, Frank; and Vietz, Daniel: Patterns For Hybrid Quantum Algorithms. In: Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2021).

Resulting Context

However, choosing f, the observables for measuring the qubits and an ansatz is not trivial. The convergence of the algorithm depends on multiple factors: the objective function and the optimization strategy for updating the parameters. Regarding the objective function, one major obstacle for the convergence of the solution are regions of the function which contain only a small norm of the gradient (these regions are also referred to as *barren plateaus*). These regions can significantly influence how fast the solution converges (Cerezo et al. 2020) or in the worst case, result in non-convergence of the algorithm. Since the optimization procedure is performed classically, the classical computation can impact the overall runtime.

Grover

Intent

How can you find a specific element (\tilde{x}) within an unsorted database of (N) elements?

Context

The goal is to find the correct bit string $\tilde{x} \in {0,1}^N$ for which the function [f: {0,1}^N \rightarrow {0,1}] [f(x) = \begin{cases} 0, & x \neq \tilde{x} \ 1, & x = \tilde{x} \end{cases}] becomes 1 (indicator function). The assumption is that the function itself is provided in the form of an oracle, and any bit string with length N can be evaluated using it.

Problem & Forces

Not available

Solution

1. **Create the superposition of all possible solutions:** [s = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle] This step involves preparing the quantum system in

an equal superposition of all possible states, representing every possible solution.

2. **Repeat the following steps until the amplitude of the desired solution $ \tilde{x}$ is "large enough":** 2.1. **Negate the amplitude of the target solution $\tilde{x}$:** Apply the oracle function that flips the sign of the amplitude corresponding to the target state (|$\tilde{x}$\rangle). 2.2. **Reflect all amplitudes about the current average amplitude:** Perform a diffusion operator (also known as the Grover diffusion or inversion about the mean), which reflects the quantum state across the average of all amplitudes. This step amplifies the amplitude of the target solution while reducing the amplitudes of other states.
3. **Measure the final quantum state:** After repeating the above steps a sufficient number of times, measure the quantum register. The result will likely be the desired solution $\tilde{x}$.

Resulting Context

The solution string $\tilde{x}$

Oracle

Also known as: *This pattern has also been referred to as Black Box.*

Intent

Re-use a computation of a quantum algorithm without necessarily knowing the implementation.

Context

A Divide-and-Conquer approach is a commonly used method that simplifies solving a complex problem.
In such an approach, an oracle can be used as an reusable part of a quantum algorithm - a black blox with hidden internals. Multiple oracles can be used as building blocks to compose larger algorithms.

Problem & Forces

Not available

Solution

The concrete implementation of an oracle is highly problem-specific. As a consequence, there are many different types of oracles, of which several are discussed in (Gilyen et al. 2019).

Resulting Context

Concrete implementations of oracles add to the overall depth to the overall quantum algorithm but are often neglected in the runtime of an oracle-based algorithm.

Quantum Associative Memory (QuAM)

Also known as: *Enter your input for this section here.*

Intent

Encode data in a compact manner that do not require calculations

Context

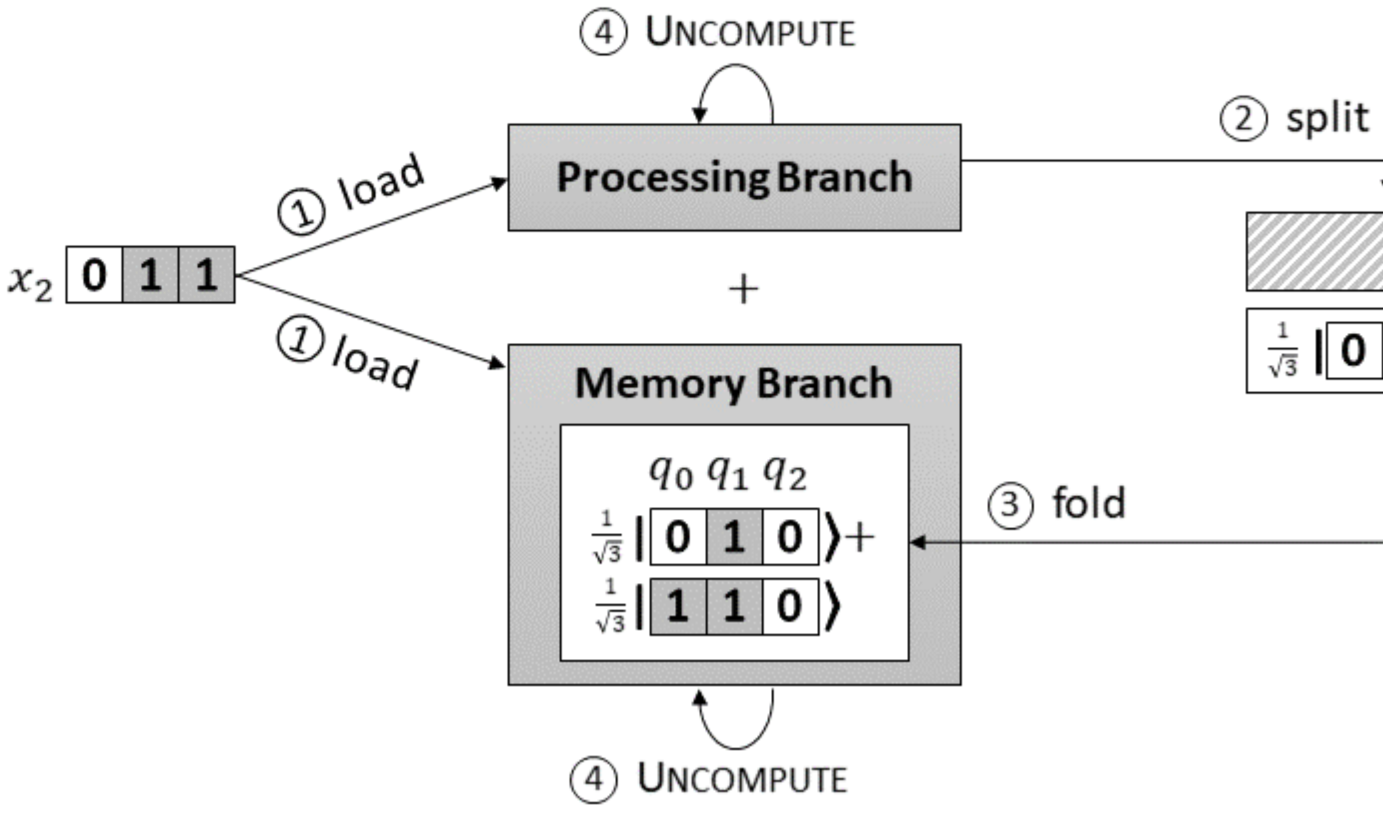
A quantum algorithm requires multiple numerical values X as input for further calculations.

Problem & Forces

Not available

Solution

Use a quantum associative memory (QuAM) to prepare a superposition of basis encoded values inthe same qubit register (Leymann and Barzen 2020a). Note that the quantum register is an equally weighted superposition of the basis encoded values. Both branches have a load and a storage part (see sketch). An additional element is first prepared into the loadpart of both branches. Next, the processing branch is split in such a manner, that the new element gets a properamplitude such that it can be brought into superposition with the already added elements. Finally, an Uncompute cleans the processing branch to be ready for the next iteration (see (Ventura and Martinez 2000) for details.)



Pattern sketch, taken from: Weigold, Manuela; Barzen, Johanna; Leymann, Frank; Salm, Marie: Data Encoding Patterns for Quantum Algorithms. In: The Hillside Group (Hrsg): Proceedings of the 27th Conference on Pattern Languages of Programs (PLoP '20).

Resulting Context

The resulting encoding is a digital encoding and therefore suitable for arithmetic computations (Leymann and Barzen 2020). For input \$n\$ numbers that are approximated by \$l\$ digits, \$l\$ qubits are needed for this representation. Each of then encoded input values is represented by a basis vector with an amplitude of1√n.All other \$2^{l–n}\$ amplitudes of the register are zero - in our example, \$|000\rangle\$, \$|001\rangle\$,\$|100\rangle\$,\$|101\rangle\$, and \$|111\rangle\$. The amplitude vector is therefore often sparse for this encoding (Schuld and Petruccione 2018).

Quantum-Classic Split

Also known as: *Enter your input for this section here.*

Intent

Split computational tasks into a quantum part running on a quantum computer and a classical part running on a classical computer.

Context

There are many quantum algorithms that require pre-processing or post-processing which must be done on a classical device. Thus, it is often necessary to split the algorithms into quantum parts and classical parts. Similarly, to run a quantum algorithm on a quantum computer that contains a small number of, possibly noisy, qubits, it may also be beneficial to split the algorithm into a quantum part of reasonable size and a classical part (Preskill 2018).

Problem & Forces

Not available

Solution

The main idea is to split an algorithm into quantum parts and classical parts. However, how an algorithm can be split depends on the problem and its implementation.

Resulting Context

Since the part running on a classical computer and the part on a quantum computer can depend on each other, e.g., in an iterative solution, several interactions between the systems may be needed. Consequently, if a classical computer and a quantum computer are connected via a queue, the queuing time, i.e., the time messages are waiting in the queue, also contributes to the overall runtime.

Quantum Random Access Memory (QRAM) Encoding

Also known as: *Enter your input for this section here.*

Intent

"Use a quantum random access memory to access a superposition of data values at once" (Weigold et al. 2021)

Context

For accessing the values of input data, a random access memory is needed.

Problem & Forces

Not available

Solution

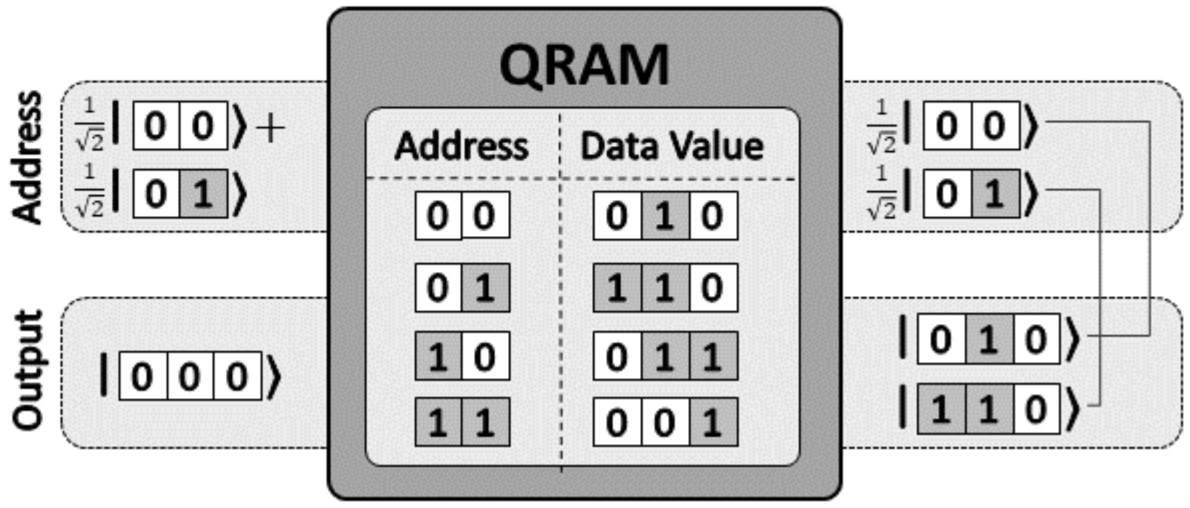
When a classical random access memory (RAM) gets an address to a memory index, it transfers the data value stored at this address into a specified output register. The functionality of quantum random access memory (QRAM) is similar, however, the registers are not classical but quantum registers (Johnston, Harrigan and Gimeno-Segovia 2019). Consequently, the address and output registers can be in superposition instead of only single values. The solution sketch demonstrates the loading process of a QRAM (Giovannetti, Lloyd and Maccone 2008) holding a superposition of the first two addresses ($\newcommand{\state}[1]{{\left| #1 \right>}} \frac{1}{\sqrt{2}}\state{00} + \frac{1}{\sqrt{2}}\state{01})$ as input register. Loading the data values of the corresponding addresses results in the following state:

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \state{\psi} = \frac{1}{\sqrt{2}} \state{00}\state{010} + \frac{1}{\sqrt{2}} \state{01}\state{110} $$

Loading m of n data values using a QRAM can be generalized as follows (Schuld and Petruccione 2018):

$$ \newcommand{\colVec}[1]{% inline column vector \bigl(\begin{smallmatrix} #1\end{smallmatrix}\bigr) } \newcommand{\bigColVec}[1]{% inline column vector \left(\begin{matrix}#1\end{matrix}\right) } \newcommand{\state}[1]{{\left| #1 \right>}} \frac{1}{\sqrt{m}}\sum_{i=0}^{m-1} \state{a}*i\state{0}* \ *\underrightarrow{QRAM}* *\frac{1}{\sqrt{m}}\sum* \state{a}_i\state{x_a} $$ }^{m-1

Thereby, the first register specifies the address register containing a superposition of m addresses, and the second register is specified as output register. Furthermore, $\newcommand{\state}[1]{{\left| #1 \right>}} \state{a}_i$ denotes the i-th data value address which has to be loaded and x_a is the data value stored at this address. The task of the QRAM is to load each data value $\newcommand{\state}[1]{{\left| #1 \right>}} \state{x_a}$ of the addresses in the first register to the output register. Consequently, $\newcommand{\state}[1]{{\left| #1 \right>}} \state{a}_i\state{x_a}$ is part of the combined output state of both registers. In dependence of the address values and their corresponding data values, this could create entanglement.



QRAM
Address
Data Value
Address
Output
$\frac{1}{\sqrt{2}}|00\rangle +$
$\frac{1}{\sqrt{2}}|01\rangle$
$|000\rangle$
00 | 010
01 | 110
10 | 011
11 | 001
$\frac{1}{\sqrt{2}}|00\rangle$
$\frac{1}{\sqrt{2}}|01\rangle$
$|010\rangle$
$|110\rangle$

Pattern sketch, taken from: Weigold, Manuela; Barzen, Johanna; Leymann, Frank; Salm, Marie: "Expanding Data Encoding Patterns For Quantum Algorithms." In: 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C), IEEE, 2021.

Resulting Context

To apply this encoding, l qubits are needed to represent the data values in Basis Encoding. For the address register, additionally $\lceil log(n) \rceil$ qubits are needed containing up to n addresses. Since \textsc{Basis Encoding} is used to represent the data values, the computational properties are similar to other digital encodings (e.g., Quantum Associative Memory (QuAM) and Basis Encoding): Since a set of data values is represented in superposition, the data values can be manipulated at once (using quantum parallelism). Furthermore, multiple arithmetic operations (e.g., addition or multiplication) for Basis Encoding are known that can also be applied to values in superposition.

Note that algorithms that specify the usage of a QRAM share the following assumption: State preparation via the QRAM is efficient and, therefore, of logarithmic runtime (Schuld and Petruccione 2018). However, to our best knowledge, there are currently no commercial hardware implementations for QRAM. Thus, a different state preparation routine has to load the data in the specified encoding. A major disadvantage today is that currently no state preparation routine for an arbitrary state is known to be as efficient as the logarithmic runtime of a QRAM. As a result, a theoretically exponential speed-up of an algorithm using QRAM is only possible if the state preparation can be realized by an efficient state preparation method.

Quantum Phase Estimation (QPE)

Also known as: *Phase estimation algorithm (PEA)*

Intent

Approximate the eigenvalue of a unitary matrix.

Context

Given a unitary matrix U and one of its eigenstates, the corresponding eigenvalue should be determined. The eigenstate $\newcommand{\state}[1]{{\left| #1 \right>}} \state{v}$ is given on a register in Basis Encoding. Applying U to the eigenstate $ \newcommand{\state}[1]{{\left| #1 \right>}}\state{v}$ results in a global phase: $

$U\newcommand{\state}[1]{{\left| #1 \right>}}\state{v} = e^{2\pi i \varphi} \state{v}$$ where the eigenvalue $\lambda = e^{2\pi i \varphi}$ is uniquely determined by $ \varphi \in [0,1]$. Therefore, it is sufficient to estimate φ.

Problem & Forces

Not available

Solution

Use the circuit shown in the pattern sketch to estimate the approximation of $ \theta$. First, a register of m ancillae is brought into an Uniform Superposition. Next, controlled versions of powers of U are applied on the register of the eigenstate following the scheme depicted inthe pattern sketch. Each application of a controlled-U operation results in a *phase kickback* of the control qubit, i.e., this qubit acquires a relative phase of φ. This results in the overall state: $$ \newcommand{\state}[1]{{\left| #1 \right>}} \state{\psi} = \sum_{y=0}^{2^m-1}e^{2\pi \varphi y}\state{y}$$ where φ is encoded in the relative phase. To extract this information, the inverse of the quantum fourier transformation is applied on the ancilla register.

Resulting Context

If φ is a rational number, the ancilla register contains the eigenvalue in Basis Encoding (assuming a proper number of anchillae). Otherwise, an approximation is produced with a probability of at least $\frac{4}{\pi}$. Increasing the precision of the approximation by adding more ancillae is costly because this also increases the number of required controlled-U operations. Because of these demanding hardware requirements, this algorithm is often regarded as non-suitable for NISQ devices.

Speedup via Verifying

Intent

Achieve a computational speedup on computation when verification of a solution is simple.

Context

There are many problems for which finding a solution is hard, but for an alleged solution it is easy to verify whether it is correct. For example, determining the factorization of a certain number is hard when the number is huge, but multiplying prime numbers to verify that their product equals the number is simple. Thus, to check if a list of prime numbers is the factorization of a specific number, multiplying the prime numbers is sufficient to verify the factorization.

Problem & Forces

Not available

Solution

For certain problems, finding a solution can be done faster by exploiting quantum parallelism: First, all possible solutions are created, then this list of solutions is searched for correct solutions by verifying the correctness of each solution. The Grover algorithm is used for searching through the possible solutions relying on an oracle to verify whether the possible solutions are correct. This means that \$O(\sqrt{N})\$ applications of the oracle are needed to determine the solution.

Resulting Context

This pattern requires that an oracle is given which can verify the solutions.