Providing Quantum Readiness: The Vision of the ProvideQ Toolbox

Domenik Eichhorn[1], Maximilian Schweikart[1], Nick Poser[1], Tobias Osborne[2], and Ina Schaefer[1]

Abstract: Quantum computing has the potential to exponentially accelerate the solution of specific problems compared to classical computing. However, the accessibility to quantum computing is currently limited due to the technical challenges posed by quantum devices. Additionally, implementing quantum algorithms is challenging because of the complex nature of quantum systems. To address these challenges, we introduce the ProvideQ Toolbox, a framework designed to enhance the accessibility of quantum computing, especially for optimization problems. Our toolbox includes a range of classical and quantum state-of-the-art optimization algorithms and employs meta-solver strategies to determine the best quantum optimization algorithm or quantum subroutine in a classical algorithm for a given optimization problem. The ProvideQ Toolbox can be used via a web-based frontend and an API and can be seamlessly integrated with multiple quantum computing backends and classical optimization frameworks.

Keywords: quantum software engineering, quantum algorithms, meta-solver strategy

1 Introduction

Quantum computing is a major topic of research since the first ideas for quantum computers were proposed in the 1980s [Be80; Fe85]. Even though no quantum hardware was available during that time, researchers were able to show that quantum computing has the potential to solve certain algorithmic problems exponentially faster than classical computing [DJ92; Sh99]. Recent advantages in quantum hardware started the so-called noisy intermediate-scale quantum (NISQ) era, which led to prototypes of noisy quantum computers and raised hope that practical quantum advantages can be achieved within the following years. However, quantum computing is still in its early stages. Quantum devices available today provide up to a few hundred logical qubits [CN23], but due to technical challenges, high maintenance costs, and high demand, their accessibility is very limited.

Quantum software development is supported by specialized programming languages, libraries, and frameworks [He20]. However, quantum software engineering generally lacks tool support and established processes [De22]. Classical software developers are used to

[1] Karlsruhe Institute of Technology, Institute of Information Security and Dependability (KASTEL), Am Fasanengarten 5, 76131 Karlsruhe, Germany, provideq@lists.kit.edu

[2] Leibniz University Hannover, Institute of Theoretical Physics, Appelstraße 2, 30167 Hannover, Germany, tobias.osborne@itp.uni-hannover.de

software engineering that provides many high-level development tools and abstraction layers, well-established processes, and other state-of-the-art practices that greatly simplify modern software development. Features like this do not yet exist in the quantum field. Implementing quantum software demands significant effort. Code optimization for underlying quantum devices is essential, and incorporating complex error correction techniques becomes necessary to mitigate the effects of decoherence and noise. Furthermore, quantum computing and quantum algorithms operate fundamentally differently from their classical counterparts. Quantum computing has the potential to highly impact aspects of the world by providing large-scale computational speedups and allowing the calculation of solutions that would not be possible with classical computers. However, its high entry barrier makes it inaccessible for most people and companies, which highly reduces the potential impact of this technology.

To alleviate this problem, we aim to improve the accessibility of quantum computing with the ProvideQ toolbox: a framework that implements a range of classical and quantum state-of-the-art optimization algorithms and acts intermediate layer that can be used to speed up combinatorial optimization problems with quantum subroutines. Furthermore, we employ so-called meta-solver strategies, which determine the best solution approach to an optimization problem by analyzing problem instances based on pre-defined heuristics.

2 Making Quantum Computing Accessible

There are many recent initiatives to improve quantum software engineering, for instance: the development of quantum SDKs such as Qiskit [Qi23], CirQ [Ci22], Forest [SCZ16], or Strawberry Fields [Ki19], the development of specialized libraries and frameworks such as Pennylane [Be18] for quantum machine learning and chemistry, QuTiP [JNN12] for quantum simulation, or Qiskit-optimization [Qi23] for solving optimization problems. There are initiatives like PlanQK[3] that focus on the deployment of quantum applications and providing knowledge exchange platforms, and some companies already work on the commercialization of hybrid solutions for optimization problems.

In the ProvideQ project we want to build on these existing initiatives and design a framework that acts as an abstraction layer when using hybrid approaches to solve optimization problems. We focus on the implementation of so-called meta-solver strategies, which combine classical and quantum algorithms to provide highly efficient solutions to optimization problems. A meta-solver strategy predicts the best solution path for a given problem instance by making assumptions about its characteristics and then selecting optimal solvers based on empirical data and expert knowledge. The proposed solution paths consist of several classical and quantum subroutines, which the ProvideQ toolbox will orchestrate and execute on suitable backends (classical computation clusters or quantum hardware). The main challenge for the ProvideQ project is identifying potentials for quantum speedups, which is why another

[3] `https://planqk.de/` Accessed: 29.08.2023

main effort of the ProvideQ toolbox is the analysis of classical state-of-the-art algorithms and gathering knowledge about the practical performance of quantum algorithms.

As practical quantum advantages will most likely only be achieved once post-NISQ quantum hardware is available, the benefits of the ProvideQ toolbox are currently mainly from a theoretical perspective. Thus, when developing the ProvideQ toolbox, we have to focus on designing a future-proof framework that can be maintained over a long period. We ensure this by addressing the following requirements:

1. Easy to Use: The toolbox should be easy to understand. It has to be well-document and easy to integrate into other tools. Inputs in the form of standardized file formats have to be supported.
2. Extendable: It should be easy for third-party developers to extend the ProvideQ toolbox with new input formats and quantum and classical subroutines.
3. Exchangeable Quantum Backends: Quantum backends constantly evolve while new quantum hardware is built. The backends on which the ProvideQ toolbox can deploy quantum code should be exchangeable.
4. Relevance: Algorithms and problems implemented by the ProvideQ toolbox should focus on relevant optimization problems and quantum algorithms.
5. Open Source: We want to establish a development community around the toolbox and provide all code (including frontend, backend, classical and quantum algorithms, and meta-solver strategies) open-source on GitHub using the MIT license.

3 The ProvideQ Toolbox

The ProvideQ toolbox is currently under development, and a first prototype is already released. The code of the project is available on GitHub[4], and the prototype can be accessed via a web interface[5] and an API[6]. The toolbox consists of two main components: the server and the web client. Figure 1 shows an overview of the architecture.

The *toolbox server* component is the heart of our framework. It implements problem definitions, meta-solver strategies, and concrete solvers. Problem definitions are representations of different optimization problems (such as MaxCut or Knapsack). They define a standardized input and output format for every algorithmic problem supported by the ProvideQ toolbox. For every problem definition, we implement one meta-solver strategy and a set of concrete solvers. Concrete solvers are implementations of classical or quantum algorithms. They can be either monolithic, standalone algorithms, or polylithic algorithms

[4] `https://github.com/ProvideQ` Accessed: 29.08.2023
[5] `https://provideq.kit.edu/` Accessed: 29.08.2023
[6] `https://api.provideq.kit.edu/` Accessed: 29.08.2023

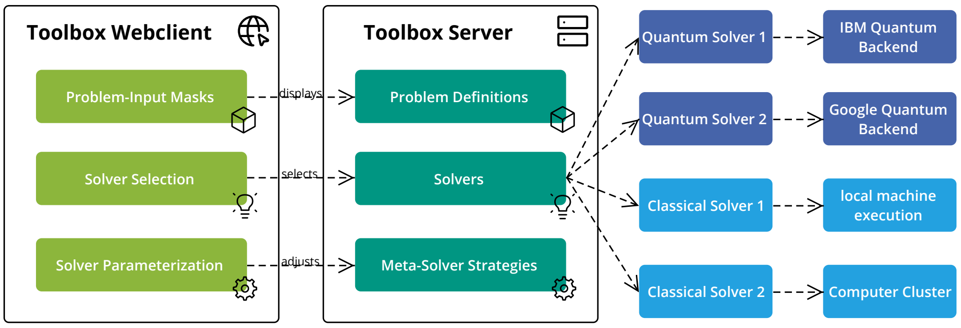

Fig. 1: Overview of the ProvideQ toolbox software architecture.

with multiple exchangeable subroutines. Meta-solver strategies are implemented as decision trees that perform multiple analyses on a given problem instance, make assumptions about its characteristics, and then iterate through the decision tree to decide which solvers should be used to solve a problem optimally. The toolbox-server component also implements multiple interfaces that (1) allow external programs to access the problem definitions, solvers, and meta-solver strategies over an API and (2) implement communication channels with classical computing networks and quantum hardware to execute the concrete solvers.

The *web client* component is one example of an external program using the API of the toolbox server. It is a web-based graphical user interface that visualizes all components of the server. A user can click on a problem definition to open a site where he can insert an instance of the associated algorithmic problem. He can then select a specific solver to calculate a solution or ask the ProvideQ toolbox to select the solver for him based on a meta-solver strategy. He can also help the ProvideQ toolbox to select an optimal solver by providing additional assumptions about the given problem instance through extra text fields, sliders, and checkboxes.

4 Conclusion and Future Work

Quantum computing has a high potential to speed up the solving of optimization problems drastically, but its true potential can only be realized when it is accessible to a broad audience. We presented the vision of the ProvideQ toolbox, a framework designed to enhance the accessibility of quantum computing by providing an abstraction layer between optimization problems and quantum computing hardware. Implementing effective meta-solver strategies and quantum subroutines within ProvideQ necessitates extensive research efforts. ProvideQ is an ongoing project, and our future work will revolve around the constant improvement of the ProvideQ toolbox. We aim to build a community around our framework and expand its capabilities by incorporating additional meta-solver strategies, hybrid algorithms, and interfaces to more quantum backends.

Acknowledgements

This work has been supported by the German Federal Ministry for Economics and Climate Action in project ProvideQ (reference number: 01MQ22006F).

References

[Be18] Bergholm, V. et al.: Pennylane: Automatic differentiation of hybrid quantum-classical computations. arXiv preprint arXiv:1811.04968/, 2018.

[Be80] Benioff, P.: The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. Journal of statistical physics 22/, pp. 563–591, 1980.

[Ci22] Cirq Developers: Cirq, version v1.1.0, 2022, URL: `https://doi.org/10.5281/zenodo.7465577`, visited on: 08/29/2023.

[CN23] Collins, H.; Nay, C.: IBM Unveils 400 Qubit-Plus Quantum Processor and Next-Generation IBM Quantum System Two, 2023, URL: `https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two`, visited on: 09/06/2023.

[De22] De Stefano, M. et al.: Software engineering for quantum programming: How far are we? Journal of Systems and Software 190/, p. 111326, 2022.

[DJ92] Deutsch, D.; Jozsa, R.: Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences 439/1907, pp. 553–558, 1992.

[Fe85] Feynman, R. P.: Quantum mechanical computers. Optics news 11/2, pp. 11–20, 1985.

[He20] Heim, B. et al.: Quantum programming languages. Nature Reviews Physics 2/12, pp. 709–722, 2020.

[JNN12] Johansson, J. R.; Nation, P. D.; Nori, F.: QuTiP: An open-source Python framework for the dynamics of open quantum systems. Computer Physics Communications 183/8, pp. 1760–1772, 2012.

[Ki19] Killoran, N. et al.: Strawberry fields: A software platform for photonic quantum computing. Quantum 3/, p. 129, 2019.

[Qi23] Qiskit contributors: Qiskit: An Open-source Framework for Quantum Computing, 2023.

[SCZ16] Smith, R. S.; Curtis, M. J.; Zeng, W. J.: A practical quantum instruction set architecture. arXiv preprint arXiv:1608.03355/, 2016.

[Sh99] Shor, P. W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review 41/2, pp. 303–332, 1999.