Quantum Foundations

1 INTRODUCTION

Quantum computing is a relatively new computational paradigm, and prototypical *quantum processing units (QPU)* have only recently become available. QPUs feature quantum bits, or *qubits*. Unlike normal bits featured by classical CPUs, qubits are not limited to only two possible states (0 or 1), but can assume states *in between* 0 and 1, using the quantum phenomena of *superposition*. Quantum states expressed by qubits are exponentially larger than those assumed by classical bits. Coupled with the possibility to exploit further quantum phenomena such as *entanglement*, these properties provide QPUs with computational advantages over CPUs. However, QPU development is still at an early stage, and consequently, QPUs are governed by various limitations, making it impossible to leverage these advantages for practical problems just yet. Nonetheless, the prospect of achieving computational speedups with QPUs is promising, in particular for highly practical fields like databases.

It is impossible to provide a complete introduction to the field here, and we refer to Nielsen and Chuang [18] as seminal textbook, and Bharti et al. [4] as algorithmic review instead. Nonetheless, since quantum computing differs greatly from the customary hardware foundations, we provide the reader with a refresher on all essential topics.

2 OVERVIEW ON QUANTUM ALGORITHMS

Existing classical algorithms cannot be trivially converted to quantum systems. Instead, quantum algorithms following specific quantum computational paradigms are required. These algorithms fall, broadly speaking, into two categories: Algorithms with provable speedups over their classical counterparts that require a fully functioning, error corrected quantum computer to work, and heuristic approaches working on hardware that is physically realisable now and in the medium term future.

The first category includes, for instance, Shor's seminal factoring algorithm, or Grover search in large, unstructured spaces.[1] The second class contains heuristic approaches that are believed to optimally use quantum phenomena in non-error corrected, present-day quantum computers, so-called noisy intermediate scale quantum (NISQ) machines. Except for notable efforts like supremacy experiments on an artificially constructed problem [3] or a kernel-based supervised machine learning approach [14], algorithms in this class do not enjoy a strict proof of computational advantage, or even currently experimentally verified advantages on available hardware. Nonetheless, seeking near-term QPU-DB utility, we are mostly concerned with this class of algorithms.

The class mostly comprises *hybrid variational quantum algorithms*: parameterised quantum gates (that we explain in detail below) produce quantum states (using superposition and entanglement) that are supposed to, roughly speaking, explore scenarios faster than classical counterparts. Detailed rationales on why speedups are expected, and the physical intuition behind, cannot be discussed here for the lack of space. Instead, we refer to Bharti *et al.* [4] for gate-based systems, and Albash and Lidar [2] for quantum annealers.

Since quantum algorithms are tied to their respective quantum computing paradigms, we next outline two such paradigms and discuss our considered quantum algorithms in detail.

[1]Grover's algorithm was initially misnamed as *database search*, albeit the meaning of database does not match what is considered a typical database: The algorithm works on *conceptual* search spaces, not physical data, unless these can be stored in quantum RAM, whose implementation in quantities even beyond a few quantum bits only is still a major physical challenge [4],

3 QPU ARCHITECTURES

While physical implementation techniques for quantum computers vary widely, two conceptual paradigms have emerged: Gate-based QPUs and quantum annealers. The following outlines their essential characteristics.

3.1 Gate-based QPUs

Multiple vendors (*e.g.*, IBM, AQT, IQM, Rigetti) offer access to early commercial hardware that follows the gate-based quantum computation model. To run on these QPUs, an algorithm needs to conform to this model. As such, we outline the gate-based computation model, and discuss a suitable quantum algorithm for solving optimisation problems like join ordering. The problems need to be encoded as a specific quadratic formulation, that we discuss next. Finally, we outline the process of embedding the algorithm onto a gate-based QPU, and describe fundamental QPU limitations that impact this embedding.

Computation Model. In the gate-based model, quantum gates form a *quantum circuit* that implements a computation, and forms part of an algorithm. Quantum gates operate (conceptually similar to electronic gates) on the state represented by a collection of qubits. Thereby, the classical state represented by the input qubits can be transformed into a quantum state that exhibits quantum properties, such as superposition and entanglement. At the end of the circuit, the created quantum state is typically measured.

Measurements (very roughly speaking [19]) collapse a quantum state into *one of multiple possible* classical states. The outcome follows a probability distribution that depends on the quantum state produced by the circuit. As such, quantum computation is inherently probabilistic, and circuits are not just run once, but instead iterate over tens to thousands of *shots* to obtain a statistical distribution. A common algorithmic pattern is to perform operations on an input state, such that measurement collapses the created quantum state into a favourable classical state, representing a solution to a problem, with high probability.

Gate-Based Quantum Optimisation. To solve join ordering on gate-based QPUs, we require a quantum algorithm for optimisation problems that follows the gate-based computation model. Our considerations for gate-based QPUs rest on the *quantum approximate optimisation algorithm* (QAOA) [9]. The overall structure of the QAOA circuit is depicted in Fig. 1. QAOA is an iteration-based, hybrid quantum-classical algorithm: Each iteration has (1) a quantum step, in which a parameterised quantum circuit explores the search space using quantum states, followed by a measurement, and (2) a

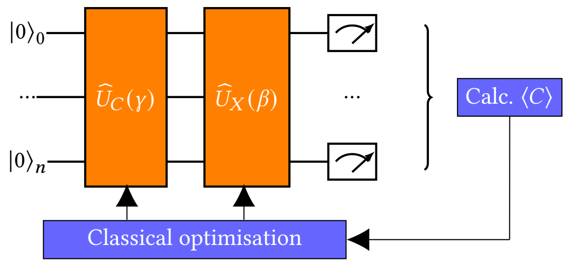

Figure 1: The overall structure of the QAOA circuit, with gates parameterised by γ and β, which are classically optimised between iterations.

classical step to tune the circuit parameters with a classical numerical optimiser, based on the measurement results. The optimised parameters are used for the next iteration. The quantum circuit consists of a repeating and alternating sequence of quantum operators. Note that these operators involve multiple partial gates that operate on each qubit or qubit pair separately. Rather than depicting each operation explicitly, Fig. 1 summarises them into larger logical operators.

An important QAOA parameter is given by the number of operator repetitions, typically denoted p. As shown by Farhi *et al.* [9], the approximation quality improves as p increases. However, even for $p = 1$, QAOA shows promising experimental results for some problems [9, 10], and it is know that an efficient classical algorithm for sampling the output distribution of QAOA with $p = 1$ would imply a collapse of the polynomial hierarchy [12], which is seen as clear indicator of possible quantum advantages. It is still an open research question if and when *practical* benefits can be obtained on current hardware. Optimisation (sub-)problems that can be mapped to QAOA appear frequently in databases, and often do not concern the payload itself[2]—instead, they address *conceptual* search spaces, for instance for join ordering, and are therefore promising candidates for quantum speedups.

Problem Encoding. QAOA requires a problem to be encoded as an *Ising Hamiltonian* [9], which is (for our purposes) equivalent to a *quadratic unconstrained binary optimization* (QUBO) problem formulation [5, 13], which (1) only allows *quadratic*, or pairwise, interactions between variables, (2) is *unconstrained*, and thereby prohibits explicit constraints, (3) only allows *binary* variables, and (4) encodes *optimisation* problems. Physically, we may interpret a QUBO problem as an *energy formula*, where we seek to determine the variable configuration corresponding to the minimum, or *ground energy state* (optimal solution). Mathematically, QUBO problems are based on the multivariate polynomial

$$f(\vec{x}) = \sum_i c_{ii}x_i + \sum_{i\neq j} c_{ij}x_i x_j, \tag{1}$$

where $x_i \in \{0, 1\}$ are variables, and $c_{ij} \in \mathbb{R}$ coefficients (it holds that $c_{ij} = c_{ji}$). Note that Eq. (1) can be interpreted as weighted, undirected graph with adjacency matrix given by the coefficients c_{ij}. The QAOA algorithm seeks to determine $\arg\min_{\vec{x}} f(\vec{x})$.

Problems must be cast into QUBO form by ensuring that the ground energy state, determined by the quantum algorithm, corresponds to a valid and optimal solution (this is possible for all NP-complete problems [15]). One particular challenge is to find a QUBO reformulation of problems that aligns well with the QPU properties. For instance, more pairwise variable interactions result in deeper QAOA circuits [9], which are therefore subject to larger degradation by current-day technical imperfections. This interplay between problem encoding and QPU architecture remains characteristic for quantum computing, and further motivates QPU-DB co-design.

QPU Embedding/Transpilation. After building the QAOA circuit based on the QUBO formulation, it must be *embedded* onto the QPU to match the hardware constraints regarding qubit topology and available gates [7]. The logical circuit is *transpiled* into a functionally equivalent circuit that only uses gate operations included in the *native QPU gateset.* Moreover, limited interaction possibilities between qubits must be accounted for by inserting additional gates [7] when non-adjacent qubits are supposed to interact with each other. This prolongs the circuit, and thereby its execution time, which is very costly, considering QPU limitations.

QPU Limitations. The execution of a physical circuit on NISQ hardware is subject to various perturbations that decrease result quality. Most importantly, execution time on QPUs is inherently limited by the coherence time [18]: Increasing quantum circuit depth (*i.e.*, the longest sequence of operators in a circuit), and thereby the execution time, increases the occurrence probability of decoherence errors because of a loss of quantum information due to interactions with the environment [21]. The probability of *gate errors* likewise increases with increasing gate count. Consequently, it is important to find formulations of limited size, limiting circuit depth and execution time. The quantum circuit depth is therefore a crucial metric for evaluating quantum computing feasibility, and consequently plays a substantial role in our considerations.

3.2 Quantum Annealers

Quantum annealers (as, for instance, offered by D-Wave [17]) implement a restricted variant of adiabatic quantum computing (the general form is known to be equivalent to gate-based quantum computing [1]) on NISQ hardware [11, 20]. It is subject to debate whether or not quantum annealing (QA) can achieve speedups over classical systems [20].

Currently available quantum annealers offer thousands of qubits, compared to hundreds of qubits in gate-based systems. They only allow for interactions between a restricted set of qubit pairs, as given by their connectivity graph.

Computation Model. QA also seeks to determine the minimum energy, or ground state, of a problem Hamiltonian [11]. The same QUBO encoding as for gate-based QPUs is applicable to quantum annealers. QA can find minimum energy solutions to QUBOs, but is incapable of running other quantum algorithms.

Contrary to gate-based systems, where computation time is determined by the depth of a circuit and the types of gates, the

[2]Quantum RAM is extremely hard to manufacture, and current technology allows for producing a few qubits at most [4], which will likely remain a challenge in the near- and mid-term future. Loading even parts of the payload into quantum resources to work quantum algorithms on the actual data is therefore not a viable path.

annealing time t_A is a parameter for annealers: If it is too small, a non-optimal solution will result; if it is too large, this is obviously counter to speedups. Again, due to the probabilistic nature of quantum computing, typically many quantum annealing runs, or shots, are performed to form a batch of results. Therefore, as a metric for evaluating the QA feasibility, we consider the time $t_O = t_A \cdot n$, where n denotes the number of shots required to obtain an optimal or near-optimal result at a high probability.

The minimally required annealing time is (roughly, and ignoring many details) inversely proportional to the minimum energy gap between the ground and first excited state of the Hamiltonian [11]. This eigenvalue gap depends on the chosen QUBO encoding, and is, as a quantum many-body problem, difficult to compute [2]. Annealing times are therefore usually subject to empirical determination.

Additionally, pairwise interactions between variables must be mapped to the possibilities of physical hardware, and too many interactions (e.g., for increasing problem dimensions) make the mapping infeasible. Consequently, similarly to gate-based QPUs, the feasibility of QA is highly dependant how well the concrete QUBO encoding aligns with the underlying quantum hardware.

QPU Embedding. To solve a QUBO problem on physical hardware, we need to map the QUBO graph onto the connectivity graph of the QA. Variables are mapped to qubits, and coefficients c_{ij} in Eq. (1) determine the physical coupling strengths between qubits. In the D-Wave Advantage system, every physical qubit is connected to a limited set of 15 other qubits following a certain topological pattern (Pegasus graph [17]). Consequently, not every possible interaction c_{ij} can be directly realised on hardware.

To represent such interactions, sets of connected physical qubits are combined into qubit chains [6] so that not directly connected qubits can be made to interact. A larger mismatch between QUBO and QA hardware graph leads to more and longer such chains. This substantially reduces the usable amount of qubits, and increases the error probability during the annealing process [17].

Embedding QUBO onto a given QPU topology was performed by hand in first applications of QA on DB problems [22], which is possible for specific problems with predictable connectivity requirements following fixed patterns (like multi-query optimisation), but becomes infeasible for more dynamic problems like join ordering. Additionally, the problem is NP-complete [16, 23], and any manual approach quickly becomes infeasible. Approximative or heuristic techniques (see, *e.g.*, Refs. [8, 23]) are available; yet, we find that reducing the complexity of the QUBO formulation remains a crucial requirement.

4 METRICS FOR QUANTUM COMPUTING FEASIBILITY

In summary, we find several metrics for evaluating the feasibility of quantum computing:

(1) For gate-based QPUs, the depth of the embedded QAOA circuit determines the execution time, and is essential w.r.t. QPU limits,
(2) For quantum annealing, we consider the overall annealing time required to obtain an optimal or near-optimal result at a high probability,
(3) In all scenarios, the qubits required by the QUBO encoding limit problem dimensions fitting on QPUs.

Circuit depth and qubit requirements heavily depend on the applied QUBO encoding. Hence, we strongly consider them for our QUBO encoding for join ordering, derive upper qubit bounds for the encoding, and apply the metrics for both, our experimental analysis of join ordering on state-of-the-art QPUs as well as for assessing improvements achievable by co-design.

REFERENCES

[1] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. 2008. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Rev.* 50, 4 (2008), 755–787.

[2] Tameem Albash and Daniel A. Lidar. 2018. Adiabatic quantum computation. *Rev. Mod. Phys.* 90 (Jan 2018), 015002. Issue 1. https://doi.org/10.1103/RevModPhys.90.015002

[3] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (01 Oct 2019), 505–510. https://doi.org/10.1038/s41586-019-1666-5

[4] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. 2022. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* 94 (Feb 2022), 015004. Issue 1. https://doi.org/10.1103/RevModPhys.94.015004

[5] Zhengbing Bian, Fabián Chudak, William Macready, and Geordie Rose. 2010. *The Ising model: Teaching an old problem new tricks.* Technical Report. D-Wave Systems Inc.

[6] P. I. Bunyk, Emile M. Hoskinson, Mark W. Johnson, Elena Tolkacheva, Fabio Altomare, Andrew J. Berkley, Richard Harris, Jeremy P. Hilton, Trevor Lanting, Anthony J. Przybysz, and Jed Whittaker. 2014. Architectural considerations in the design of a superconducting quantum annealing processor. *IEEE Transactions on Applied Superconductivity* 24, 4 (April 2014), 1–10.

[7] Lukas Burgholzer, Rudy Raymond, and Robert Wille. 2020. Verifying results of the IBM Qiskit quantum circuit compilation flow. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Denver, CO, USA, 356–365.

[8] D-Wave Systems Inc. 2022. Minorminer library for embedding Ising problems onto quantum annealers. https://docs.ocean.dwavesys.com/en/stable/docs_minorminer/source/intro.html

[9] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. (Nov. 2014). arXiv:1411.4028

[10] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. (Dec. 2014). arXiv:1412.6062

[11] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. 2000. Quantum computation by adiabatic evolution. (Jan. 2000). arXiv:quant-ph/0001106

[12] Edward Farhi and Aram W Harrow. 2016. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. https://doi.org/10.48550/arxiv.1602.07674 arXiv:1602.07674

[13] Mark Lewis and Fred Glover. 2017. Quadratic Unconstrained Binary Optimization Problem Preprocessing: Theory and Empirical Analysis. (May 2017). arXiv:1705.09844

[14] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. 2021. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics* 17, 9 (01 Sep 2021), 1013–1017. https://doi.org/10.1038/s41567-021-01287-z

[15] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014), 5.

[16] Jirí Matousek and Robin Thomas. 1992. On the complexity of finding iso- and other morphisms for partial k-trees. *Discret. Math.* 108 (1992), 343–364.

[17] Catherine McGeoch and Pau Farré. 2020. *The D-Wave Advantage system: An overview.* Technical Report 14-1049A-A. D-Wave Systems Inc.

[18] Michael A. Nielsen, Isaac Chuang, and Lov K. Grover. 2002. Quantum computation and quantum information. *American Journal of Physics* 70, 5 (April 2002), 558–559.

[19] Asher Peres and Leslie E. Ballentine. 1995. Quantum Theory: Concepts and Methods. *American Journal of Physics* 63, 3 (1995), 285–286. https://doi.org/10.1119/1.17946

[20] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.

[21] Eleanor Rieffel and Wolfgang Polak. 2011. *Quantum computing: A gentle introduction.* MIT Press, Cambridge, MA.

[22] Immanuel Trummer and Christoph Koch. 2016. Multiple query optimization on the D-Wave 2X adiabatic quantum computer. *Proceedings of the VLDB Endowment* 9, 9 (May 2016), 648–659.

[23] Stefanie Zbinden, Andreas Bärtschi, Hristo Djidjev, and Stephan Eidenbenz. 2020. Embedding algorithms for quantum annealers with Chimera and Pegasus connection topologies. In *High Performance Computing*. Springer International Publishing, Cham, 187–206.