Quantum Singular Value Transformation (QSVT) in Qiskit

Qiskit Hackathon Europe - Research Study Groups
April 20 - June 14 2021

Bartu Bisgin, Nagme Oruz, Jiří Guth Jarkovský, Erfan Abedi and Martin Mauser

I. Motivation

Quantum Singular Value Transformation (QSVT) is a relatively new and promising framework for gate-based quantum computation, which allows for efficient circuit representations of a large class of polynomial transformations, including non-unitary operations [1]. This significantly expands the potential range of NISQ-era applications and quantum algorithms in general. QSVT also serves as a unification scheme for many quantum algorithms by showing that they can all be expressed in terms of the same circuit structure [1], [2]. The simple universal circuit structure that realizes the polynomial transformations generally uses only a constant number of ancilla qubits, regardless of the size of the input. Given the previous works on Quantum Signal Processing (QSP) [3], it also has the potential to result in novel algorithms.

Despite being such a potentially groundbreaking framework, it remains relatively unknown and unexplored outside academia. Our work focuses on making QSVT available to a broader community, by implementing a readily usable QSVT module for Qiskit. The simplest QSVT-implementable algorithm is Quantum Search, which we have decided, can serve as an intuitive introduction to the framework. Whereas the original Grover's uses oracles that flip the phases on the marked elements (requiring a different oracle to mark each element); the QSVT search oracle performs search via applying a polynomial transformation on the inner product of the initial and target states which maps it to the value of 1. This means the initial and target states completely overlap; hence, the search is done! Our QSVT search is based on the Fixed-Point Amplitude Amplification [4].

This work offers a QSVT Search algorithm that does not require implementing different oracles for both different number of qubits and different marked elements as in the original Grover's algorithm, meaning it makes search in higher dimensions almost trivial. Our code can find any marked element up to 9 qubits efficiently using one circuit structure and merely increasing the depth of the circuit. It should be noted that the structure of generic Grover's beyond 4 qubits is not accessible to the broader Qiskit community explicitly. Our implementation also employs the so-called Fixed-point Amplitude Amplification scheme, which utilizes the whole Bloch sphere to perform the search instead of the mere 2D plane employed in original Grover's, resulting in a much more efficient search which always converges, avoiding undercooking and overcooking (the souffle problem) [2], [4].

The details of the mathematical theory behind QSVT has been given in the documentation of the code instead of here for purposes of clarity and conciseness.

II. The Algorithm

The basic tool for implementing polynomial transformations, and hence, non-unitary quantum operations is Block-encoding [1], [5]. Block-encoding describes the embedding of a non-unitary matrix of interest A into the upper-left block of a larger unitary matrix U_A. The circuit then, is expressed in terms of U_A. What QSVT allows is the evaluation of $f(A)$ where $F(x)$ is a smooth, real or complex-valued function, and $f(x)$ is the polynomial approximation of $F(x)$. In general, this procedure generates an approximation to a matrix function, which is a central goal in numerous computational tasks. This includes many quantum operators such as e^{-iHt} (for Hamiltonian Simulation [6]) and A^{-1} for matrix inversion for solving systems of equations (HHL algorithm [7]).

The procedure behind QSVT then is to approximate $F(x)$ via a quantum signal processing routine. In a nutshell, for any polynomial approximation $f_d(x) \approx F(x)$ of degree d (where $F(x)$ is a smooth function that satisfies the QSP constraints such as having a definite parity and being absolute-value bounded by 1 between $x = [-1, 1]$), there exists a set of unique phase factors $\Phi \in (-\pi, \pi]^d$ which can then used inside a series of R_Z gates on some ancillary qubit, interleaved with the block-encoded Unitary U_A

in the circuit to yield the transformation $f(A)$ [1], [5]. It must be noted that in general the procedure of finding these phase factors is not trivial, but there has been some recent progress which we make use of [5].

In our case, the matrix A is actually a scalar (1×1 non-unitary matrix) which is the inner product $\langle s|t\rangle$, where $|s\rangle$ is the initial state and $|t\rangle$ is the target state. This is the easiest block-encoding one can achieve. Then, a polynomial approximation $f_d(x) \approx F(x)$, where $F(x)$ is close to a step function can transform the inner product from its initial value (which is greater than 0) to the value of 1, which would be the complete overlap with the target state, meaning that the search is done!

The algorithmic details of how any QSVT algorithm would be implemented in Qiskit practically is as follows:

1) Find an approximation to the function of choice in terms of Chebyshev polynomials, obtain coefficients of this Chebyshev-Fourier expansion.
2) Using non-zero, definite parity coefficients, find the QSP phase factors via a QSP-solver algorithm (There are currently only 3 algorithms for finding these phases and we've made use of a recent library by — which employs optimization).
3) Find the circuit elements, and how they can be implemented in Qiskit for the QSVT framework (even though the circuit structure is the same for many Quantum algorithms, the interior of the gates used in each algorithms differs).
4) Build the whole circuit out of the elements.
5) Apply relevant scaling/slicing of the QSP phase factors according to the Qiskit gate definitions (for example scaling all phases by a factor of 2 because the $R_Z(\theta) = fn(\frac{\theta}{2})$). Feed the scaled/sliced phase factors into the circuit.
6) Run the circuit and run simulations in the qasm_simulator to check whether the probabilities correspond to the transformation you wanted, where probabilities are roughly $|f(A)|^2$. When A is not a scalar, this is not very straight-forward.

The biggest challenge among all of these are extracting the phase factors and building up the correct block-encoding circuit. Most of our effort went into translating theory into implementation. The circuit structure of the general QSVT can be seen in Fig.1.

III. Implementation

For performing search within the QSVT scheme, one should encode $A = \langle s|t\rangle$ into a Unitary U_A which is a legitimate gate that can be used within a circuit and then build the circuit that performs the transformation $f(A) \rightarrow 1$. Here, $U_A = \Pi_l A \Pi_r$ where Π_l and Π_r are the projectors onto the left and right singular spaces respectively.

With QSVT in general, one can perform arbitrary polynomial transformations that satisfy the QSP constraints, and it would be ideal to choose $f(x) \approx tanh(100x)$ as our polynomial which is close to a step function that also preserves smoothness and performs the mapping $f(x) \overset{\approx}{\rightarrow} 1$. In practicality, to find the QSP phases, this function is expanded in terms of the Chebyshev polynomials of the first kind such that $f_d(x) = c_0T_0(x) + c_1T_1(x) + ... + c_dT_d(x)$, where d is the highest power in the expansion (ie. the degree of the polynomial).

However, we are yet to find a way for feeding the QSP phases that are obtained by the QSP_solver modules in the repository for arbitrary functions into Qiskit with the correct scaling and slicing. Thus, we have limited our current circuit implementation to pure Chebyshev polynomials because the QSP phases corresponding to these have an analytic form and are very straight-forward to implement. That is, $f_d(x) = T_d(x)$ with phases $\Phi = [(1-d)\frac{\pi}{2}, \frac{\pi}{2}, ..., \frac{\pi}{2}]$.

This construction requires us to find for every $x_n = \frac{1}{\sqrt{2^n}}$ (where n is the number of qubits to perform the search upon), a corresponding Chebyshev polynomial T_d, such that $T_d\ (x = x_n) \approx 1$. We have found out that after 9 qubits, the efficiency of this construction drops greatly due to the increased circuit depth as d scales rapidly with n. However, this does serve as a good *proof of principle* for QSVT in Qiskit as we can experiment around to check for Chebyshev polynomials that do not correspond to the value of 1, and see that we can indeed perform arbitrary polynomial transformations. This has been shown in the documentation explicitly.

Taking into account that we do not require an expansion in terms of Chebyshev as we already are using pure Chebyshev polynomials and that we do not require solving the phases via a routine because they are already known, we can eliminate the steps 1) and 2) from the guidelines illustrated in Section II. Here's how we went forward with this project:

1) First, we needed to find the circuit elements U_A, Π_l and Π_r. This effort was mainly made by Bartu and Jiří in the team, and it was Jiří that finally cracked it.
2) Secondly, we needed to actually build the circuit, this was prototyped by Jiří on 2 qubits and generalized by Bartu for n qubits. This circuit by default takes $|t\rangle = |1\rangle^{\otimes n}$, and performs the mapping $f : |s\rangle \rightarrow |t\rangle$. An additional routine has been added at the end of the circuit such that $|s\rangle$ can be mapped into any valid state $|t\rangle \in [0,1]^d$ given a bit string of

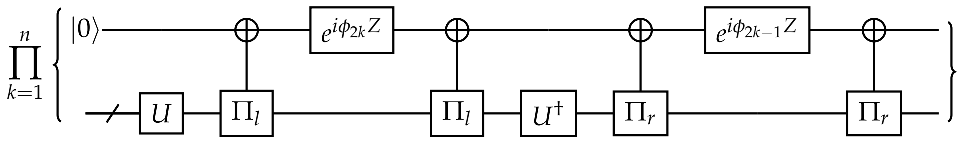

Figure 1: Circuit structure of the general QSVT algorithm, with $n = \frac{d}{2}$.

the marked element.

3) Then, the code was refactored and polished by Erfan. He was also in charge of translating the QSP_solver module of [5], [8], from Matlab and Julia into Python, which is the pack that will become very important once we figure out how to implement the obtained phases for general polynomials into our circuit.
4) Martin and Uzay have performed tests on different cases and reported bugs to be fixed. Martin also played a big role in the preparation of this paper.
5) The documentation has been prepared by Bartu and improved upon mainly by Erfan. The notes for improvement have been outlined by a team effort.
6) The contribution format has been ensured by Erfan.

IV. Project Results

We have created a simple QSVT module for Quantum search that can find the target state in a very straightforward and fundamentally different manner. Our scheme does not require different oracles for different number of qubits and different elements while generalizing very intuitively to higher dimensions.

Our results also showcase to the broader community of Qiskit users, the proof of principle for performing arbitrary transformations within quantum circuits, opening the doors for experimentation and exploration.

All of our detailed explanations, demonstrations and numerical results can be found in the doc file under the GitHub repository. We have chosen to limit all the intricacies into the documentation to keep this paper as clear and concise as possible, and also to ensure the coherency of the results in the context of the code and how it was constructed in Qiskit.

V. Impact & Outlook

This module is a first step towards a bigger library that will contain many QSVT-implemented algorithms (Search, Simulation, Factoring ...) that are readily usable by the community to probe deeper into the details of QSVT and how it might become useful in the near and long term. With this project, we hope that we have bridged the gap between academia and the Qiskit community for QSVT.

As stated, our search module work only with pure Chebyshev polynomials for now, and the challenge of building a circuit that will correctly register phases for arbitrary transformations remains. This will be the first issue to be addressed post-hackathon. We also want to incorporate the routine for finding arbitrary elements into the Oracle itself. With these solved, we will be able to move on to the other algorithms that we want to add to the library.

We believe that even in this most infant form, QSVT-Search serves as an introductory point towards QSVT for the Qiskit community, and already straightforwardly gives access to any marked state in the space of up to 9 qubits without having to resort to myriads of different oracles to solve the problem as in the Original Grover's. We hope the module does a good job of demonstrating the power of QSVT-implemented algorithms compared to their generic versions, and we are excited to see what future lies ahead.

References

[1] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. Phoenix AZ USA: ACM, Jun. 2019, pp. 193–204. [Online]. Available: https://dl.acm.org/doi/10.1145/3313276.3316366

[2] Institute for Quantum Computing, "Isaac Chuang - Grand unification of quantum algorithms," Nov. 2020. [Online]. Available: https://www.youtube.com/watch?v=GFRojXdrVXI&t=2035s

[3] G. H. Low, T. J. Yoder, and I. L. Chuang, "The methodology of resonant equiangular composite quantum gates," *Physical Review X*, vol. 6, no. 4, p. 041067, Dec. 2016, arXiv: 1603.03996. [Online]. Available: http://arxiv.org/abs/1603.03996

[4] T. J. Yoder, G. H. Low, and I. L. Chuang, "Fixed-point quantum search with an optimal number of queries," *Physical Review Letters*, vol. 113, no. 21, p. 210501, Nov. 2014, arXiv: 1409.3305. [Online]. Available: http://arxiv.org/abs/1409.3305

[5] Y. Dong, X. Meng, K. B. Whaley, and L. Lin, "Efficient Phase Factor Evaluation in Quantum Signal Processing," *Physical Review A*, vol. 103, no. 4, p. 042419, Apr. 2021, arXiv: 2002.11649. [Online]. Available: http://arxiv.org/abs/2002.11649

[6] G. H. Low and I. L. Chuang, "Optimal Hamiltonian Simulation by Quantum Signal Processing," *Physical Review Letters*, vol. 118, no. 1, p. 010501, Jan. 2017, arXiv: 1606.02685. [Online]. Available: http://arxiv.org/abs/1606.02685

[7] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for solving linear systems of equations," *Physical Review Letters*, vol. 103, no. 15, p. 150502, Oct. 2009, arXiv: 0811.3171. [Online]. Available: http://arxiv.org/abs/0811.3171

[8] qsppack, "qsppack/QSPPACK," Mar. 2021, original-date: 2020-02-24T08:11:30Z. [Online]. Available: https://github.com/qsppack/QSPPACK