---

title: 'QAOA.jl: Automatically differentiable Quantum Approximate Optimization Algorithm' tags: - Julia - quantum algorithms - automatic differentiation - optimization authors: - name: Tim Bode orcid: 0000-0001-8280-3891 equal-contrib: true affiliation: 1 affiliations: - name: Institute for Quantum Computing Analytics (PGI-12), Forschungszentrum Jülich, 52425 Jülich, Germany index: 1 date: 19 January 2023 bibliography: paper.bib

Summary

Quantum algorithms are an area of intensive research thanks to their potential of speeding up certain specific tasks exponentially. However, for the time being, high error rates on the existing hardware realizations preclude the application of many algorithms that are based on the assumption of fault-tolerant quantum computation. On such *noisy intermediate-scale quantum* (NISQ) devices (Preskill 2018), the exploration of the potential of *heuristic* quantum algorithms has attracted much interest. A leading candidate for solving combinatorial optimization problems is the so-called *quantum approximate optimization algorithm* (QAOA) (Farhi, Goldstone, and Gutmann 2014). `QAOA.jl` is a `Julia` package that implements the QAOA to enable the efficient classical simulation typically required in research on the topic. It is based on `Yao.jl` (Luo et al. 2019), (Luo et al. 2023) and `Zygote.jl` (Innes et al. 2019), (Innes et al. 2023), making it both fast and automatically differentiable, thus enabling gradient-based optimization. A number of common optimization problems such as MaxCut, the minimum vertex-cover problem, the Sherrington-Kirkpatrick model, and the partition problem are pre-implemented to facilitate scientific benchmarking.

Statement of need

The demonstration of quantum advantage for a real-world problem is yet outstanding. Identifying such a problem and performing the actual demonstration on existing hardware will not be possible without intensive classical simulations. This makes a fast and versatile implementation of the QAOA rather desirable. As shown in Fig. 1, `QAOA.jl` is faster in this respect than `PennyLane` (Bergholm et al. 2018), its main competitor in automatically differentiable QAOA implementations.

Mathematics

The cost function of the QAOA for a general quadratic optimization problem is typically defined as

$$\hat{C} = \sum_{i=1}^{N} \left[h_i + \sum_{j>i} J_{ij} \hat{Z}_j \right] \hat{Z}_i,$$

where the h_i, J_{ij} are real numbers encoding the problem in question, and $\hat{Z}_{i,j}$ denote Pauli matrices. Similarly, the conventional *mixer* or *driver* of the QAOA is given by

$$\hat{D} = \sum_{i=1}^{N} \hat{X}_i,$$

where the $\hat{X}_i$ are again Pauli matrices. We also introduce the initial quantum state

$$|\psi_0\rangle = |+\rangle_1 \otimes \cdots \otimes |+\rangle_N.$$

Note that this is the maximum-energy eigenstate of the driver $\hat{D}$ since $\langle\psi_0|\hat{D}|\psi_0\rangle = N$. With these prerequisites, the variational quantum state of the QAOA becomes

$$|\psi(\boldsymbol{\beta},\boldsymbol{\gamma})\rangle = \exp\left(-\mathrm{i}\beta_p\hat{D}\right)\exp\left(-\mathrm{i}\gamma_p\hat{C}\right)\cdots\exp\left(-\mathrm{i}\beta_1\hat{D}\right)\exp\left(-\mathrm{i}\gamma_1\hat{C}\right)|\psi_0\rangle.$$

The goal is then to *maximize* the expectation value

$$E_p(\boldsymbol{\beta},\boldsymbol{\gamma}) = \langle\psi(\boldsymbol{\beta},\boldsymbol{\gamma})|\hat{C}|\psi(\boldsymbol{\beta},\boldsymbol{\gamma})\rangle$$

over the variational parameters $\boldsymbol{\beta},\boldsymbol{\gamma}$. Note that `QAOA.jl` furthermore supports others drivers, e.g.

$$\hat{D} = \sum_{(i,j)\in\mathcal{E}} \left(\hat{X}_i\hat{X}_j + \hat{Y}_i\hat{Y}_j\right),$$

where $\mathcal{E}$ is the set of connections or *edges* defined by the coupling matrix J_{ij}.

References

Bergholm, Ville, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M. Sohaib Alam, et al. 2018. "PennyLane: Automatic Differentiation of Hybrid Quantum-Classical Computations." arXiv. https://doi.org/10.48550/arxiv.1811.04968.

Farhi, Edward, Jeffrey Goldstone, and Sam Gutmann. 2014. "A Quantum Approximate Optimization Algorithm." *ArXiv e-Prints*.

Innes, Mike, Alan Edelman, Keno Fischer, Chris Rackauckas, Elliot Saba, Viral B Shah, and Will Tebbutt. 2019. "A Differentiable Programming System to Bridge Machine Learning and Scientific Computing." arXiv. https://doi.org/10.48550/arxiv.1907.07587.

———. 2023. "Zygote.jl." *GitHub Repository*. GitHub. https://github.com/FluxML/Zygote.jl.

Luo, Xiu-Zhe, Jin-Guo Liu, Pan Zhang, and Lei Wang. 2019. "Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design." *ArXiv e-Prints*.

———. 2023. "Yao.jl." *GitHub Repository*. GitHub. https://github.com/QuantumBFS/Yao.jl.

Preskill, John. 2018. "Quantum Computing in the NISQ Era and Beyond." *Quantum* 2 (August): 79. https://doi.org/10.22331/q-2018-08-06-79.