The Journal of Open Source Software

VeriQuEST.jl: Emulating quantum verification with QuEST

Jonathan Miller [1*], **Cica Gustiani**[2*], **Dominik Leichtle**[2], **and Elham Kashefi**[2]

1 School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom **2** Laboratoire d'Informatique de Paris 6, CNRS, Sorbonne Université, 4 Place Jussieu, Paris 75005, France * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Verification of delegated quantum computations is a challenging task in both theory and implementation. To address the theory, methods and protocols have been developed that ouline abstract verification. Implementation will likely require a quantum network in place for certain protocols. In the mean time, specialised emulators have been developed to perform quantum computation, offering a possibility to explore verifcation numerically. Many emulators rely solely on the gate base model and do not allow for projective, mid-circuit measurements, a key component in most quantum verification protocols. In response, we present the Julia package, `RobustBlindVerification.jl` (RBV). RBV aims to emulate blind measurement based quantum computing (MBQC and UBQC) with interactive verfication in place. Quantum computation is emulated in RBV with the Julia package `QuEST.jl`, which in turn is a wrapper package, `QuEST_jll`, developed with `BinaryBuilder.jl` (Contributors, 2022) to reproducibally call the `C` library, `QuEST` (Jones et al., 2019). RBV is developed based on the work by Leichtle et al. (2021), herein referred to as 'the protocol'. The protocol is an example of robust blind quantum computation (RBVQC). It is a formal verification protocol with minimal overhead, beyond computational repetition and resistant to constant noise whilst mainting security.

Statement of need

There are many quantum computing paradigms, notably the gate or circuit based model is the most popular [**Cite**]. It is limited by most hardware providers not capable of performing mid-circuit, adpative and projective measurements [**Cite**]. The measurement-base quantum computing (MBQC) paradigm conversely is predicated on this very capability [**Cite**]. It turns out that MBQC can utilise projective measurements to offer secure delegated QC over a quantum network between clients and servers. This leads to the need for efficient, secure and verifiable delegated access. Trust in the security, data usages, compuation and algorithm implementation is not a given for delegated QC. Many protocols have been implemented to address these issues. Advancements in verification has relied on verification assuming only uncorrelated noise (Gheorghiu et al., 2018), verification assuming reliable state preparation per qubit (Kapourniotis & Datta, 2019), verification requiring more than one server and entanglement distillation (Morimae & Fujii, 2013) or verification with the assumption that a verifier has access to post-quantum cryptography unbeakable by a quantum prover (Mahadev, 2022). Such results fall short of a robust verification protocol that does not suffer from costly process or are inflexible to noise. To respond to these shortcomings and to address the problem for bounded-error quantum polynomial (BQP) computations a verification protocol is implemented(Leichtle et al., 2021). It is known that the complexity class BQP can efficiently solve binary descision problems with quantum computers. Further, the protocol is robust to constant noise and maintain security.

The basis for this verification protocol is universal blind quantum computation (UBQC), which extends meaurement based quantum computation. Commonly, a client with minimal quantum capabilities, maybe only state preparation or only measurement is theorised, and an all powerful conceptual server is connected to this client over a quantum network. MBQC works by updating the basis angle qubits are measured in by the outcomes of previous qubits. By keeping a set of secret basis angles which are incorpoarated into measurement basis, the client can perform its quantum computation free from the server being able to ascertain any informatoin [**CHECK**]. So a qubit is initialised by the client as $|+_\theta\rangle$, where θ is the rotational angle, but there is another angle, δ such that when the basis for measurment is updated, the server is told to measure with $\tilde{\theta} + \delta + r\pi$, where $\tilde{\theta}$ is the updated angle based on previous measurement outcomes and r is a one time pad random bit used to help correct measurements [**CITE BLIND**]. To turn UBQC into robust verification quantum computation (RVBQC), the protocol calls for the use of multiple rounds to run the computation along with some test rounds.

The protocol is designed such that verification is separated into the execution of N rounds, C rounds are the algorithm to be run (e.g., the computation round) and $T = N - C$ test rounds. Test rounds contain traps which can detect malicious or noisy behaviour of the server. The algorithm for test rounds has the same underlying stucture save for state initialisations and the method of adaptive basis updates. After N rounds, a classical analysis is conducted and a result computed whether the server and/or the computation were to be trusted. The computation round is prepared and executed with UBQC, whereas the test rounds utilise a trapification strategy to conduct tests against the server. The test rounds use a strategy that splits some qubits into traps and some into dummies. The traps and the dummies are prepared according to some randomness, which though the UBQC will have determinsitic outcomes that can me tested. For each test round the traps and the dummies are compared such that all trap qubit outcomes must pass a verification equation, if any one trap fails the whole test round fails. The outcome to these tests for each round are aggregated. The aggregate count of test rounds that passed the test must exceed a predetermined amount, based on parameters of the protocol. The mode repsonse for the computation round must be greater than half the number of computation rounds. The results of the test and computation rounds dictate the trust of the server. RVBQC implements the protocol under these requirements. For an in depth understanding see @Leichtle et al. (2021).

Core features and functionality

There are three core features. Firstly, the user can simply run standard MBQC if they so choose. Secondly, the user can specify to use UBQC. Thirdly, the user can run the verification protocol. Since MBQC and UBQC are universal [**CITE**] the outcomes are the same. If a user wants to explore with noise models, then the RVBQC is designed to seemlessly offer these by specifying the noise model in the server struct (e.g., `NoisyServer(model)` for `model` being the specified noise model). Julia is a transparent programming language and all functionality is available to the user. If one wishes to become acquainted with the details see the public GitHub repository for `VeriQuEST`.

Future plans

The concept of a client and server in the quantum sense requires a quantum internet. There are means to simulate a single Hilbert space for the client and the server such that the client can initialise a single qubit in the space and through Bell entanglement teleport qubit states to select qubits in the server. The server does not know these state due to the entanglement. That has not been incorporated here, instead the client initalises all qubits in the state in its own density matrix or state vector, then the server duplicates the state. The client state is no longer considered. The server state is acted upon with noise, entanglement and measurements. Here we restrict what information the client sends to the server to keep the emulation of

blindness in tact. A future work will be to introduce this client-server algorithm to better emulate the relationship.

The noise models used in this package are standard models, but may not accurately capture hardware noise realistically. To address this whole quantum state Kraus maps, density mixing, double qubit and more custom single qubit models are being developed.

Acknowledgements

We acknowledge contributions from QSL, NQCC,...???

References

Contributors, B. (2022). BinaryBuilder.jl. In *GitHub repository*. https://github.com/JuliaPackaging/BinaryBuilder.jl; GitHub.

Gheorghiu, A., Hoban, M. J., & Kashefi, E. (2018). A simple protocol for fault tolerant verification of quantum computation. *Quantum Science and Technology*, *4*(1), 015009. https://doi.org/10.1088/2058-9565/aaeeb3

Jones, T., Brown, A., Bush, I., & Benjamin, S. C. (2019). QuEST and high performance simulation of quantum computers. *Scientific Reports*, *9*(1), 10736. https://doi.org/10.1038/s41598-019-47174-9

Kapourniotis, T., & Datta, A. (2019). Nonadaptive fault-tolerant verification of quantum supremacy with noise. *Quantum*, *3*, 164. https://doi.org/10.22331/q-2019-07-12-164

Leichtle, D., Music, L., Kashefi, E., & Ollivier, H. (2021). Verifying BQP computations on noisy devices with minimal overhead. *PRX Quantum*, *2*, 040302. https://doi.org/10.1103/PRXQuantum.2.040302

Mahadev, U. (2022). Classical verification of quantum computations. *SIAM Journal on Computing*, *51*(4), 1172–1229. https://doi.org/10.1137/20M1371828

Morimae, T., & Fujii, K. (2013). Secure entanglement distillation for double-server blind quantum computation. *Phys. Rev. Lett.*, *111*, 020502. https://doi.org/10.1103/PhysRevLett.111.020502