VeriQuEST.jl: Emulating quantum verification with QuEST

29 Jan 2024

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` [@BinaryBuilder2022] to reproducibally call the `C` library, `QuEST` [@QuESTJones2019]. RBV is developed based on the work by @PRXQuantum.2.040302, 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

Quantum computing appears to be on a course that leads to the need for efficient, secure and verifiable delegated access through some client-server relationship. The so-called client-server paradigm in use for current classical computation via cloud or high performance computing is the current analogue. 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_2019], verification assuming reliable state preparation per qubit [@Kapourniotis2019nonadaptivefault], verification requiring more than one server and entanglement distillation [@MorimaeFujii2013] or verification with the assumption that a verifier has access to post-quantum cryptography unbeakable by a quantum prover [@Mahadev2022ClassicalVerification]. 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, the protocol addresses the problem for bounded-error quantum polynomial (BQP) computations [@PRXQuantum.2.040302]. 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 protocol is designed such that verification is separated into the execution of rounds. The content of each round has its own requirements. After some specified number of rounds, a classical analysis is conducted and a result computed whether the server and/or the computation were to be trusted. Each round is either a round of the required computation, called a *computation* round, or the round is used to test the server, called a *test* round. 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. 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. RBV implements the protocol under these requirements. For an in depth understanding see @@PRXQuantum.2.040302.

Core features and functionality

Core features include MBQC and UBQC capability along with running the RVBQC algorithm. An ideal setting can be used with pure state vectors or density matrices. Uncorrelated noise along with single qubit noise models can be called. *Client and server blindness is implemented through bell-state teleportation from a single qubit repeatedly re-used to the corresponding qubit on a different section of the quantum state* (**In italics as this is still being implemented**).

Acknowledgements

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

References