GHZ distillation protocols in the presence of decoherence

Sébastian de Bone*
QuTech, Delft University of Technology,
Lorentzweg 1, 2628 CJ Delft, The Netherlands
QuSoft, CWI, Science Park 123, 1098 XG
Amsterdam, The Netherlands

David Elkouss
QuTech, Delft University of Technology,
Lorentzweg 1, 2628 CJ Delft, The Netherlands
Networked Quantum Devices Unit,
Okinawa Institute of Science and Technology
Graduate University,
Onna-son, Okinawa 904-0495, Japan

ABSTRACT

In this paper, we introduce a novel heuristic approach designed to optimize the performance of Greenberger-Horne-Zeilinger (GHZ) creation and distillation protocols under decoherence. Our methodology converts these protocols into a practical set of instructions, demonstrating, through simulations, the production of higher-quality GHZ states than previously known protocols. This advancement contributes to the field of distributed quantum computing by addressing the need for high-quality entanglement required for operations between different quantum computers.

Keywords

Entanglement distillation, Decoherence, GHZ states

1. INTRODUCTION

In a *distributed* or *modular* quantum computer we connect small or medium-sized quantum computers with entangled states to emulate a large-scale quantum computer. The subsystems of such a computer could be mounted on a photonic chip or form the nodes of a quantum network. These subsystems could, *e.g.*, be ion traps, neutral atoms, and diamond defect centers, like nitrogen-vacancy (NV) , silicon-vacancy (SiV) , or tin-vacancy (SnV) centers. The main advantage of the distributed approach is scalability. The main disadvantage is the need for high-quality entanglement necessary to perform operations between different quantum computers.

For example, if we want to combine a distributed quantum computer with an error-correction code, we require entangled states to perform the syndrome measurements used to detect errors. The fundamental resource for such a system is the *Greenberger-Horne-Zeilinger* (GHZ) state:

$$|\mathrm{GHZ}_N\rangle = \frac{|0\rangle^{\otimes N} + |1\rangle^{\otimes N}}{\sqrt{2}}. \tag{1}$$

For the standard distributed surface code, with each data qubit in a different quantum computer, we require $|\mathrm{GHZ}_4\rangle$ states. In such a system, one GHZ state is generated and consumed per parity and measurement cycle. Nickerson *et al.* [6, 5] introduced protocols to generate GHZ states in this context. In the absence of decoherence, these protocols produce high-quality GHZ states. However, their large number of distillation steps makes them less useful in practical devices with limited coherence times [1]. Therefore, there is a need for GHZ protocols that perform better in realistic scenarios.

Abstract GHZ protocols given by so-called fusion and distillation operations (see Section 2) can be carried out in multiple ways. In particular, the ordering of the operations can have a strong effect on performance in the presence of decoherence. Here, we present a method for converting GHZ distillation protocols into a set of ordered instructions. We couple this method with a dynamic program for GHZ creation [2] and show that, in the presence of decoherence, the resulting GHZ states have higher fidelities than previously known protocols.

2. BINARY TREE PROTOCOLS

We consider GHZ creation protocols over N parties in a network: the "network nodes" $\{\nu^{(i)}\}_{i=1}^{N}$. The protocols have the form of a directed *binary tree*. See Fig. 1a for a graphical depiction of the binary tree corresponding to the Modicum protocol [2]. At the top of the binary tree, we find the operation that creates the final GHZ state. At each non-leave node there is either a fusion or a distillation operation and the leaves of the tree, we find the "elementary links" $\{e_i\}_{i=1}^{K}$. These are the Bell pairs created between two of the network nodes. The elementary links do not have children.

The fusion operations $\{f_i\}_{i=1}^{F}$ and distillation operations $\{d_i\}_{i=1}^{D}$ in the tree each have two children. A fusion operation creates a state $|\mathrm{GHZ}_{N_1+N_2-1}\rangle$ out of two children $|\mathrm{GHZ}_{N_1}\rangle$ and $|\mathrm{GHZ}_{N_2}\rangle$. This is possible if there is one network node that holds a qubit from both $|\mathrm{GHZ}_{N_1}\rangle$ and $|\mathrm{GHZ}_{N_2}\rangle$. A distillation operation, on the other hand, uses an entangled state of the form $|\mathrm{GHZ}_{N_1}\rangle$ to non-locally measure a *stabilizer operator* of weight N_1 from $\mathcal{S}_{N_2}$ on a target state $|\mathrm{GHZ}_{N_2}\rangle$ [3]. Here, the stabilizer operators of $|\mathrm{GHZ}_{N_2}\rangle$ are all operators

$$\mathcal{S}_{N_2} = \langle X_1X_2\ldots X_{N_2}, Z_1Z_2, Z_2Z_3, \ldots, Z_{N_2-1}Z_{N_2}\rangle. \tag{2}$$

We consider distillation as a probabilistic operation that only succeeds if the measurement outcome is $(-1)^m = +1$. In case of an outcome $(-1)^m = -1$, we reapply all operations in sub-tree of the operation $d \in \{d_i\}_{i=1}^{D}$—*i.e.*, d itself and all operations below it. More information about the specific operations applied for fusion and distillation can be found in Ref. [2].

Each binary tree satisfies $D + F = K - 1$, where D, F and K are the number of distillation operations, fusion op-

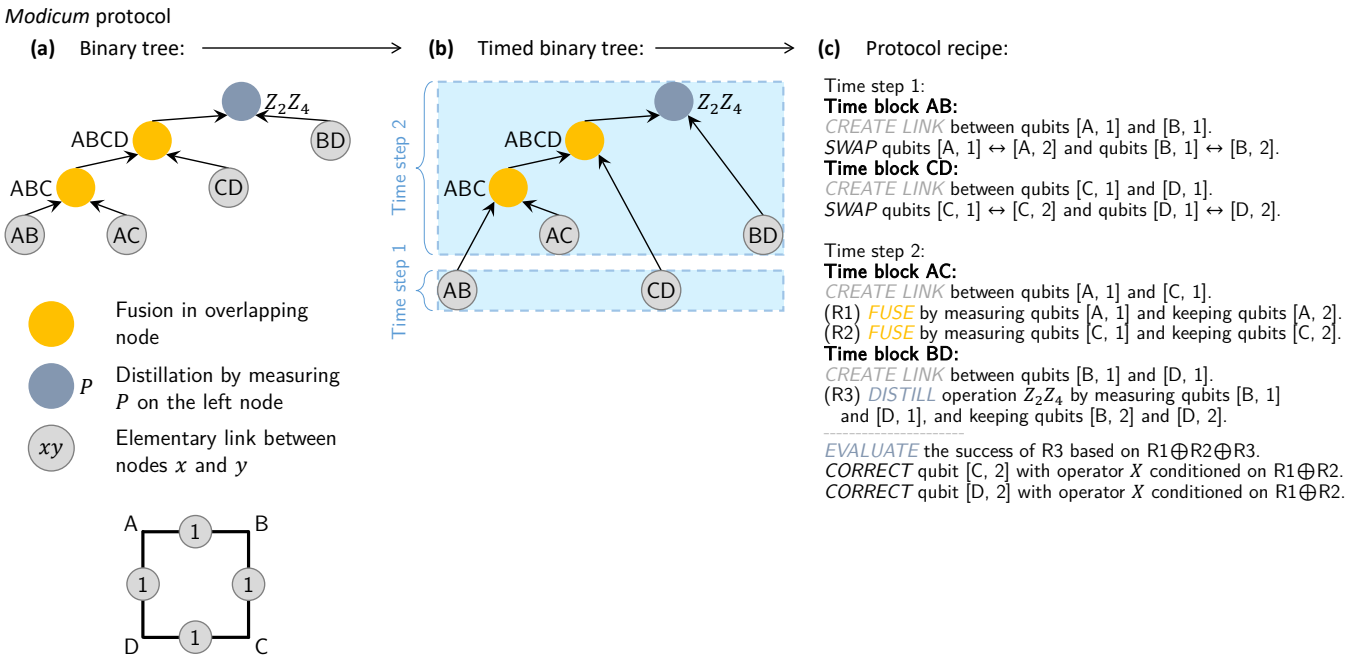

Figure 1: (a) Binary tree with $K = 4$ found with the dynamic program of Ref. [2]. (b) Identified time steps for the operations in this binary tree. (c) Identified protocol recipe for this binary tree.

erations, and elementary links, respectively. Since creating a weight-N GHZ state requires a minimum of $N-2$ fusion operations working on $N-1$ elementary links, all additional fusion operations create entangled states used for distillation purposes. Therefore, we use K (the number of elementary links in a binary tree) as a proxy for the amount of distillation that takes place in a GHZ protocol.

3. PROTOCOL RECIPE CONSTRUCTION

In the presence of decoherence, the order in which the operations are applied can have a strong impact on performance. Therefore, we convert the binary tree of a GHZ creation protocol to a "protocol recipe". This is a set of instructions that describe what specific operations have to be applied on what qubits of the network nodes. In this section, we include a condensed version of the algorithm we use to create a protocol recipe. In Fig. 1, we depict the translation of a protocol given by a binary tree (a) into a set of time steps (b) and finally into a protocol recipe (c).

For the construction of a protocol recipe, we make several physically inspired assumptions. For example, we assume a single "communication" qubit $c^{(j)}$ per node that can be used to perform operations, and a fixed number of memory qubits. We also assume that a node can only be involved in one entanglement operation at a time, and entanglement can only be created between two communication qubits.

The first step of the protocol recipe construction consists in ordering the generation of the elementary links of the binary tree. For this, we follow a recursive approach. We start at the top of the tree. At each step, we select the sub-tree with the largest size, choosing the left one in case of ties. When we reach an elementary link at one of the leaves of the tree, we add this link to an ordered list with elementary links, together with all other non-overlapping elementary links that can be carried out simultaneously. For each elementary link e that we add to the list, we check if the parent operation p of e can be applied. A parent can be applied if both of its children are contained in the list of operations. If p can be applied, we add p to the list after e. Then we check if the parent of p can be applied; if it can be added, we add it after p, *etc.* Moreover, for each operation, we associate a set of instructions that are physically possible with the target hardware. For instance, in our model, elementary links can only be created between the communication qubits, and there is only one communication qubit per node. For this reason, SWAP gates are necessary to free up the communication qubits.

Once all operations are listed, we schedule them. The schedule of operations consists of a series of time steps, with each time step divided into different time blocks. The time blocks are created such that they can be executed in parallel in different, non-overlapping parts of the network. We schedule the operations by looping over the list of operations, and, for each operation o, we check if o can be placed in an existing time block. This is possible if the network nodes in which o operates overlap with the network nodes in which the other operations of the time block operate. If it is not possible to add o to an existing time block, we create a new one. At the end of each time step, we add all required fusion corrections, as well as distillation operations that need to be evaluated at the end of the time step.

4. PROTOCOL RECIPE SIMULATION

In this section, we sketch the logic for the efficient simulation of a protocol recipe in the form of Sec. 3.

The GHZ creation protocols considered are probabilistic. To avoid situations in which they do not finish within the coherence time, we impose a *cut-off time* after which the GHZ

protocol is aborted. To be able to deal with decoherence and the cut-off time, we assign an internal time variable to each network node. At the end of each time step, we synchronize the time in all nodes. This corresponds to the protocol only moving to the next time step when all operations in the current time step are finished.

When a distillation operation fails during protocol execution, we store the time t_{fail} at which the measurement result was known, and enter "reconstruction mode". In reconstruction mode, we (re)apply all operations in this time step until t_{fail} is reached. We then move back to an earlier stage in the protocol recipe execution to reconstruct the failed state(s). As soon as we have successfully reconstructed these states, we proceed with the rest of the protocol.

5. RESULTS AND DISCUSSION

In this section, we simulate three protocol recipes in the form of Sec. 3 by executing them according to the algorithm described in Sec. 4. We execute these protocols using the open source simulator [4].

These three protocols all create GHZ states between $N = 4$ network nodes. This is arguably the most interesting configuration given its relevance for measuring the parities of a distributed surface code. In Fig. 2, we show the average GHZ fidelity and completion time of these protocols for the hardware models and parameters of the "isotopically purified" NV center sample of Ref. [1]. We show the results for the state-of-the-art success probability per entanglement attempt 10^{-4}, as well as a boosted success probability of 10^{-2}. On top of that, we show results for the Plain and Modicum protocols from Ref. [1], the Expedient protocol from Ref. [6] and the Basic protocol from Ref. [5]. We note that the Modicum protocol is also found with the dynamic program of Ref. [2]. We observe that the protocols from the dynamic program coupled with the recipe construction heuristic outperform prior protocols in the parameter regimes considered.

In future work, it is important to investigate whether the increased GHZ fidelities that we report in Fig. 2 are sufficient to operate the distributed surface code below its noise threshold. We will also optimize over protocols created with the dynamic program of Ref. [2] by making use of the algorithms described in Secs. 3 and 4 and the simulator of Ref. [1].

We note that both the dynamic program from Ref. [2] as the methods described in this paper can be applied to a general number of network nodes N. This opens the door for simulating distributed error-correction codes beyond the (weight-4) surface code.

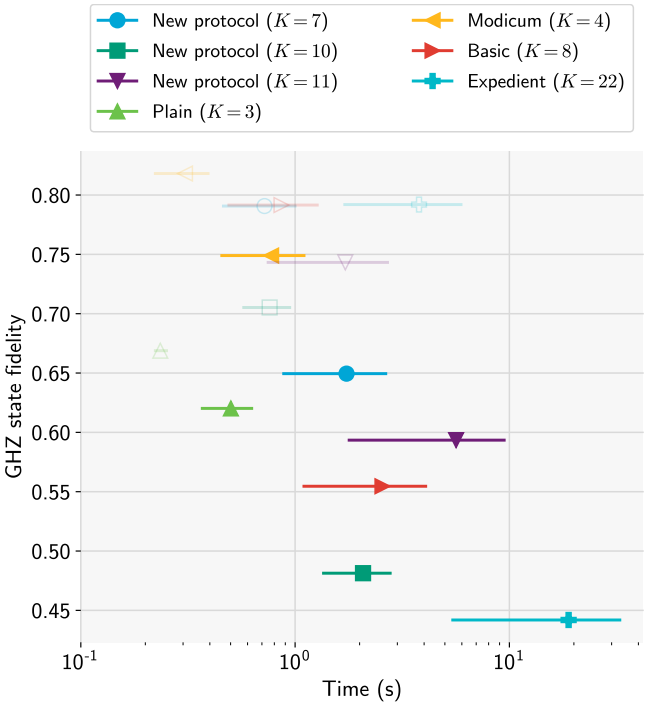

Figure 2: Average completion time and average GHZ state fidelity for a selection of GHZ creation protocols with $N = 4$, using the circuit simulator, hardware models, and parameter values from Ref. [1]. Full markers correspond to a success probability of 10^{-4} per Bell pair entanglement attempt, whereas open (and faded) markers correspond to a boosted success probability of 10^{-2} per attempt. Horizontal error bars show the distribution of the completion time (68.2% confidence interval). Each data point is based on a total of 10^5 Monte Carlo simulations, except for Expedient at 10^{-4} success probability, which is calculated with 10^4 Monte Carlo simulations.

6. REFERENCES

[1] Bradley, C. E., de Bone, S. W., Möller, P. F. W., Baier, S., Degen, M. J., Loenen, S. J. H., Bartling, H. P., Markham, M., Twitchen, D. J., Hanson, R., Elkouss, D., and Taminiau, T. H. Robust quantum-network memory based on spin qubits in isotopically engineered diamond. *npj Quantum Inf 8*, 1 (Oct. 2022), 1–9.

[2] de Bone, S., Ouyang, R., Goodenough, K., and Elkouss, D. Protocols for Creating and Distilling Multipartite GHZ States With Bell Pairs. *IEEE Transactions on Quantum Engineering 1* (2020), 1–10.

[3] Goyal, K., McCauley, A., and Raussendorf, R. Purification of large bicolorable graph states. *Phys. Rev. A 74*, 3 (Sept. 2006), 032318.

[4] Möller, P., de Bone, S., and Elkouss, D. Circuit simulator. https://doi.org/10.4121/16887658.v3, 2022.

[5] Nickerson, N. H., Fitzsimons, J. F., and Benjamin, S. C. Freely Scalable Quantum Technologies Using Cells of 5-to-50 Qubits with Very Lossy and Noisy Photonic Links. *Phys. Rev. X 4*, 4 (Dec. 2014), 041041.

[6] Nickerson, N. H., Li, Y., and Benjamin, S. C. Topological quantum computing with a very noisy network and local error rates approaching one percent. *Nature Communications 4* (2013), 1756.