Delft University of Technology

Master's Thesis

Quantum Error Correction of Patch-Loss of the Surface Code

Author
S. Hu

Supervisor
D. Elkouss

August 15, 2019

Chapter 1

Preliminary report

Introduction

Quantum computing has the potential to transcend the information technology as we know it. Small scale quantum systems are already possible today and the goal is to scale up these quantum architectures to build practical quantum devices. One approach to do this is to by networking many simple processor cells together through quantum links, avoiding the necessity to build a single complex structure. Processor cells that are located physically close to each other are connected by "short" links and lie in a patch. Patches that are located physically far from each other can in turn be connected by "long" links, such as remote optical connections. The total state of system, which contains the stored information, is shared across these patches, such that it can be accessed in either one of these patches.

This is somewhat analogous to the idea of a shared database. Many online services that we use today rely on servers that host the data that we want to view, store or edit. This data is often not stored on a single server, but copied to many others, in a shared database. In case one of these servers goes offline due to file corruption or an electricity outrage, the data is not lost, and can still be accessed on another server in the cluster.

In our Quantum network, information cannot be copied across different processor cells due to the no cloning theorem. In stead, it is shared across cells through entanglement. A cell can also go "offline", when a qubit or multiple qubits are lost from the system due to some interaction with the environment. This process is called decoherence, also described with *loss* or *erasure*. Luckily, if the losses are not too much, these cells can be restored through quantum error correction (QEC) such that the quantum state or encoded information can still be extracted from the system.

Quantum errors

Errors that can occur during Quantum computation can generally be classified as 1) noise, in which there is an error are within the computational basis, or as 2) a loss, in which the qubit is taken out of the computational basis. Losses are both detectable and locatable, which means that a higher rate of loss (p_{loss}) can be tolerated than noise or computational errors (p_{com}). The process of finding and correcting these errors is called decoding.

Kitaev's surfaces codes are defined by a set of stabilizers which act on a set of physical qubits that lie on the edges of a square lattice [1]. The stabilizers commute, and are generated by plaquettes (group of Z operators), or by stars (group of X operators). Logical operates corresponds to a set of stabilizer operators along a homologically nontrivial cycle. Any homologically equivalent set of operators can be used to measure the physical qubit operator. Therefore, in the case of a qubit loss, another set of operators can be used, if there is no *percolated* region of losses that span the entire lattice.

To decode for computational errors, one measures the stabilizer generators, which returns eigenvalue -1 on the edges of the error chains or syndromes, and can be corrected by finding a nontrivial closing chain, which either equals a stabilizer measurement that corrects the error, or a logical operator which equals a logical error. This problem is equivalent to the two dimensional random-bond Ising model (RBIM), where the shortest path needs to be found between matching pairs. The closing chain is found using the Edmonds' minimum weight perfect matching (MWPM) algorithm. This algorithm scales quadratic in time as the lattice size increases [2]. More recently, an almost-linear decoding approach has been described by Delfosse et al. [3].

There are also multiple methods to decode for losses on the surface code. Stace et. al [2, 4] describes the method of so-called superplaquettes and superstars, in which the lost qubit is accounted for by combining neighboring stabilizers. The resulting lattice can be than decoded using the same MWPM algorithm. Delfosse et al [5] describes a linear-time maximum likelihood method to decode for losses. Here, the errors are found in a *peeling* algorithm that iteratively peels branches away from a tree of possible error chains until the lost qubits remain.

Patch loss

In terms of the surface code, a patch is a set of neighboring qubit cells on the lattice that lie physically close to each other. The entire lattice is built by connecting multiple patches. It is possible that cells within the same patch suffer the same decoherence, due to their physical vicinity, such that entire patches may be lost. The Quantum links that connect these patches, in turn, may suffer a larger amount of noise due to the distance it must cross.

These patch losses may be investigated using the decoding algorithms described above. There are two main question to be answered here. The first one involves the size of the patches. What are tolerable sizes of patches such that in the event of a patch loss, the encoded Quantum information can still be extracted from the system? The second question concerns the Quantum links between these patches. What effects does the patch model on the surface have on the tolerable noise thresholds, especially on the thresholds of the Quantum links that connect different patches?

Project

This project will focus on the two problems of the patch model described in the previous section. We will use simulations to calculate for the thresholds on patch sizes and noise. Most probably, we will try to build this simulation on top of the fault-tolerance simulations by N. Nickerson [6], whose thesis will also be used as our foundation on quantum error correction. Furthermore, the introduction on topological codes will be provided by the lecture notes of D. Browne [7]. We we use the knowledge acquired in the courses Advanced Quantum Mechanics (AP3051G), Computational Physics (AP3082D), as well as the EDX courses The Building Blocks of a Quantum Computer

(DelftX - QTM2x, QTM3x) and Quantum Cryptography (CaltechDelftX - QuCryptox), which are similar to the TU Delft courses AP3421 and CS4090.

The project will be done under supervision of David Elkouss, who leads the Elkouss group, which is part of the Roadmap Quantum Internet and Networked Computing at QuTech. Weekly meetings are planned for discussions and evaluations. A presentation for the group will be held after a month. Furthermore, we also expect to be working closely together with Sebiastian de Bone, who is a Phd student at the Elkouss Group.

The project can be divided into 5 successive objectives:

1. Theory and prediction of model with patch-loss of the surface code.
2. Calculation of threshold on patch sizes.
3. Definitions of noise between patches.
4. Combine patch-loss and noise between patches.
5. Improve decoder.

The preparation for the master thesis ends on April 12, 2019. From here, the first 3 months of the project, until July 2019, we expect to be working on objective 1, expanding our theoretical knowledge of the Quantum erasure code and formulating definitions of the patch model. Ideally, we will also have some simulations results for objective 2 by the end of July. We will take a short break in August, whereafter we will continue on objectives 3 and 4. Objective 5, improving the decoder, is an optional objective. The mechanism of the decoders are so complicated, that an improvement may be a project on its own. However, due to my particular interest in these decoders, I will be alert on trying to find a point of improvement if possible. The timeline of this project will look as such.

If the simulation of the patch sizes (2) will be so complicated such that after it cannot be completed before the end of July, we will drop objectives 3 and 4 and focus on the first two objectives. The deadline for the thesis is set to the end of November, such that a defense may be held in December.

Chapter 2

Introduction

Bell states

There are four maximally entangled two-qubit Bell states. Together, they form a maximally entangled basis known as the Bell basis.

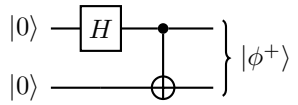
$$\left|\phi^{+}\right\rangle = \frac{1}{2}\left(\left|00\right\rangle_{AB} + \left|11\right\rangle_{AB} \right) \tag{2.1}$$

$$\left|\phi^{-}\right\rangle = \frac{1}{2}\left(\left|00\right\rangle_{AB} - \left|11\right\rangle_{AB} \right) \tag{2.2}$$

$$\left|\psi^{+}\right\rangle = \frac{1}{2}\left(\left|01\right\rangle_{AB} + \left|10\right\rangle_{AB} \right) \tag{2.3}$$

$$\left|\psi^{-}\right\rangle = \frac{1}{2}\left(\left|01\right\rangle_{AB} - \left|10\right\rangle_{AB} \right) \tag{2.4}$$

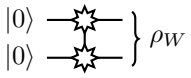
The Bell state that we want to prepare, $|\phi^{+}\rangle$, can be created by applying the Hadamard and the CNOT gate on two qubits.



However, due to the fact that the system is imperfect, a Werner state will be created instead with fidelity F, which is given by the following density matrix,

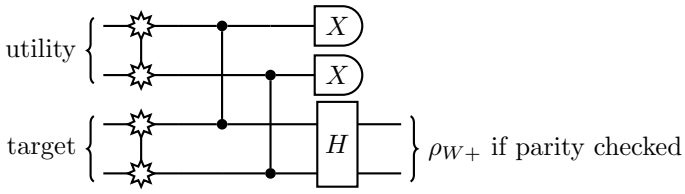
$$\rho_W = F\left|\phi\right\rangle\left\langle\phi\right| + \frac{1-F}{3}\sum_{i=1,2,3}\left|\psi_i\right\rangle\left\langle\psi_i\right| \tag{2.5}$$

where $|\phi\rangle$ now represents the Bell pair that we want to prepare, and $|\psi_i\rangle$ are the other three Bell states. The production of this noisy Bell pair will be illustrated by

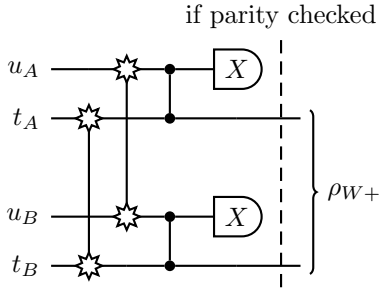


Single selection

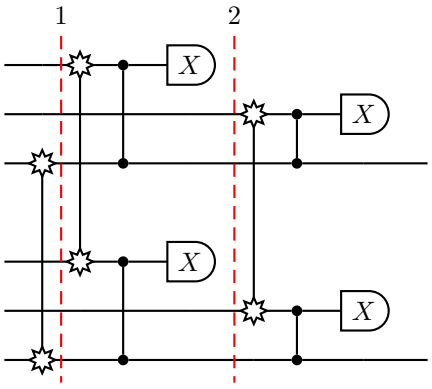
Luckily, the fidelity of a noisy Bell pair can be enhanced through entanglement purification. In the process of *single selection*, a single noisy Bell pair is used (utility pair) in order to enhance the fidelity of another noisy bell pair (target pair). Here CZ gates are applied between the first and second qubits of the two noisy pairs. A successful parity check on the utility pair yields an enhanced target pair.



We should separate the qubits based on their location. Alice and Bob both holds one of the two qubits of both the utility and target pair. We put the qubits held by Alice on top, and Bob on the bottom.



The enhanced Bell pair $\rho_{W,1}$ can be iteratively enhanced using the same scheme using newly created noisy Bell pairs ρ_W. After repeated iterations the fidelity will converge to some maximally attainable value, dependant on the fidelity of the noisy Bell pair. Two iterations of single selection is displayed below.



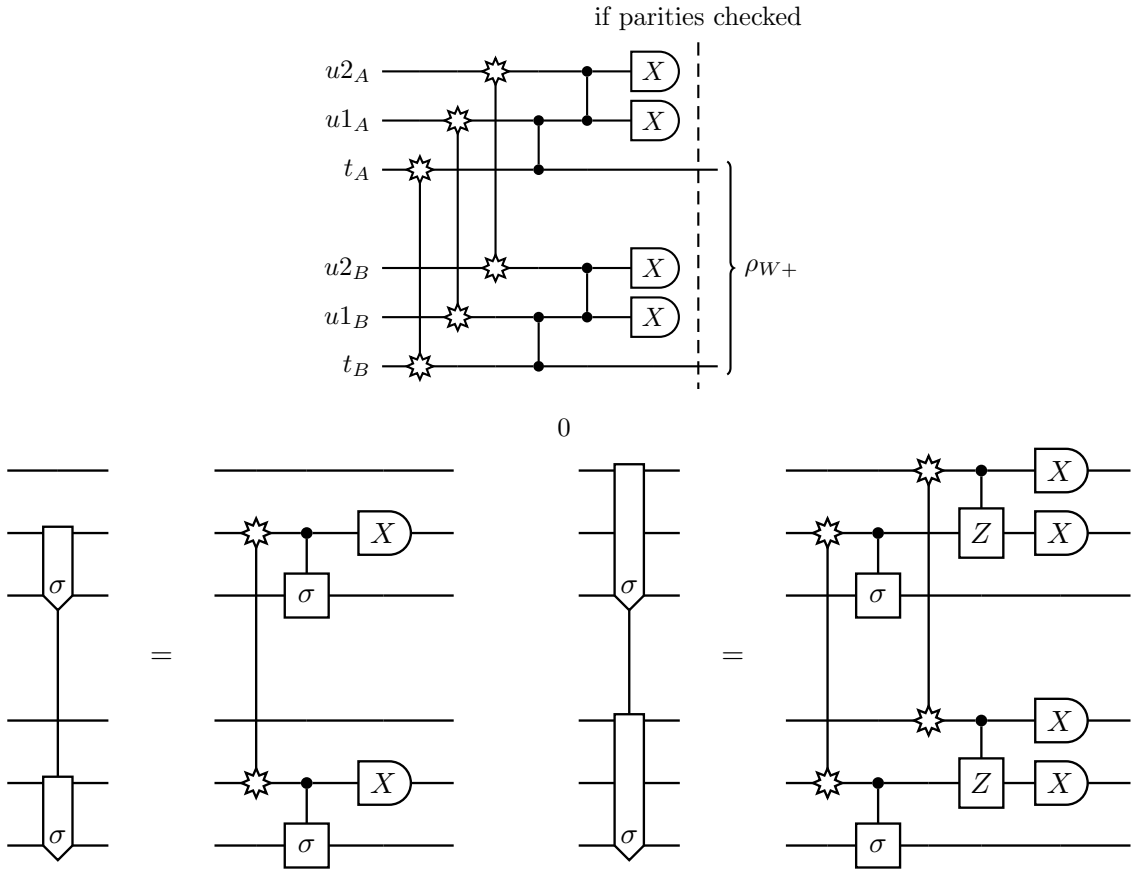
Entanglement pumping

Within each iteration, we can add additional *nested levels* to further improve the purification. In each nested level, we perform extra rounds of single selection, but taking two enhanced Bell pairs

from the previous level as input. For example, in the first nested level, a Bell pair ρ_{W+} is used to purify another Bell pair with the same fidelity to ρ_{W++}, which will be the input state for the second nested level.

Double selection

Another approach which yields a higher enhanced fidelity utilizes two noisy Bell pairs ($u1$ and $u2$) to enhance a third pair (t). They perform a CZ gate locally between $u1$ and t, followed by another between $u2$ and $u1$. Now, they perform parity checks on $u1$ and $u2$. If both parity checks are passed, the protocol succeeds and the target qubit fidelity is enhanced.



Chapter 3

The toric code

3.1 Decoders

3.1.1 The optimal decoder

3.1.2 Minimum Weight Perfect Matching

3.1.3 The Union-Find decoder

The fastest known algorithm for the MWPM problem still has a time complexity of $O(|A|\sqrt{|V|}) = O(n^2\sqrt{n})$ where n is the number of qubits. To reduce this time complexity, an algorithm has been developed with a time complexity of $O(n\alpha(n))$ [1], where α is an inverse Ackermann function, which is smaller than 3 for any practical input size (see the appendix for the exact definition). The algorithm in this thesis is the weighted growth version of the decoder described in [1], as this has a higher threshold than the original version. It should be noted that it has small differences with the algorithm fully described in [1], as the weighted growth version is not fully described. The name is a reference to the main part of the algorithm, the use of the Union-Find data-structure algorithm developed by Tarjan [5].

To get an almost linear time complexity, the syndrome is first reduced to an erasure syndrome, which can be decoded in linear time. First of all, the peeling decoder for the erasure syndrome will be explained.

	scales with n	scales with (dependant on) p
number of clusters	✓	increases for small p
size of clusters	only for large p	✓
number of cluster growths	✓	✓
number of merging events	✓	increases for small p
number of child clusters	only for large p	increases for small p

Peeling algorithm: main components

Data: AnLoc = anyon locations, ErLoc = erasure locations
Result: Graph object containing all the Clusters

```
n  for Vertex in AnLoc:
       if Vertex not ∈ Graph:
           add Vertex to Graph
       if Vertex ∉ Clusters:
           add new Cluster
1          Neighbors = FindVertexNeighbors(Vertex, Cluster) (Al. 5)
+n         while Neighbors ≠ None: /* a maximum of n extra vertices are added on initial loop    */
               for Vertex in Neighbors:
1                  new Neighbors = FindVertexNeighbors(Vertex, Cluster)
```

Complexity: O(n)

Algorithm 1: FindClusters

```
   Data: Graph containing even and uneven clusters
   Result: Graph containing only even clusters
n  select Cluster with SelectGrowClusters (Al. 6)
↓  while uneven Clusters ∈ Graph:
n      for Cluster in selection: /* total cluster growths scales with n                    */
           for Boundary (Edge, near Vertex, far Vertex) in Cluster:
               if near HalfEdge ∉ Clusters:
                   /* edge not grown, first growth-step                                   */
                   Cluster ← near HalfEdge
                   if far HalfEdge ∈ Clusters:
                       Cluster ← Edge
n                      ParentCluster = GetParentCluster(far Edge) (Al. 7)
                       if ParentCluster = Cluster:
                           RemList ← Boundary
                       else:
                           MergeList ← (Cluster, ParentCluster)
                   else:
                       no action, edge is grown a half-step
               else:
                   /* edge is already half-grown, second growth-step                      */
                   Cluster ← HalfEdge
                   if far Vertex ∉ Clusters:
                       Cluster ← Vertex
                       NewBoundaryList ← far Vertex
                   else:
                       ParentCluster = GetParentCluster(far Vertex)
                       if ParentCluster ≠ Cluster:
                           MergeList ← (Cluster, ParentCluster)
n          for Boundary in RemList: /* dependant on number of merges                     */
               /* these boundaries are excluded from second growth-step                  */
               remove Boundary from Cluster
1          for Vertex in NewBoundaryList: /* dependant on cluster size                   */
               /* vertices lie on far end of completely grown edge, this is the new boundary */
               find new Boundary with FindVertexNeighbors (Al. 5)
n      for Cluster 1, Cluster 2) in MergeList:
n          ParentCluster 1 = GetParentCluster(Cluster 1)
n          ParentCluster 2 = GetParentCluster(Cluster 2)
           if not already merged:
               Cluster 1 ∪ Cluster 2
       select grow clusters with SelectGrowClusters
   Complexity: O(n^2)
```

Algorithm 2: GrowClusters

```
   Data: Graph containing even clusters
   Result: Graph containing clusters of trees
n  for Cluster in Graph:
       get random Vertex from Cluster
       mark Vertex as traversed
       NewNodes ← Vertex
↓      while NewNodes ≠ None:
1          for Vertex in NewNodes: /* dependant on cluster size                    */
               get Neighbors (edges, vertices) of Vertex ∈ Cluster
               for neighbor Vertex, Edge) in Neighbors:
                   if neighbor Vertex is traversed:
                       remove neighbor edge from Cluster
                   else:
                       mark neighbor Vertex as traversed
                       NewNodes ← neighbor Vertex
   Complexity: O(n)
```

Algorithm 3: TraverseTrees

```
   Data: Graph containing clusters of trees
   Result: Graph containing the matching, syndrome edges
n  for Cluster in Graph:
↓      for Vertex in Cluster:
1          while # connected Edge = 1: /* dependant on cluster size               */
               remove Edge from Cluster
               if Vertex = anyon:
                   matching ← Edge
                   remove anyon status of Vertex
                   flip anyon status of opposite Vertex
               opposite Vertex is new Vertex
   Complexity: O(n)
```

Algorithm 4: PeelTrees

Peeling algorithm: functions

```
   Data: Vertex, Cluster
   Result: Neighbors of input Vertex within the Cluster
1  for neighboring Vertex, Edge of input Vertex:
   |  if Vertex ∉ Graph: /* newly found vertices and edges                    */
   |  |  Graph ← Vertex, Edge
   |  |  if Vertex ∈ AnLoc:
   |  |  |  Vertex is an anyon
   |  else:
   |  |  if Edge ∉ Graph: /* vertex already added but edge not                */
   |  |  |  Graph ← Edge
   |  if Edge ∈ ErLoc:
   |  |  add Vertex, Edge to Cluster
   |  |  if Vertex ∉ a Cluster:
   |  |  |  Neighbors ← Vertex
   |  else:
   |  |  Boundary of Cluster ← (Vertex, Edge)
   Complexity: O(1)
```

Algorithm 5: FindVertexNeighbors

```
   Data: Graph
   Result: list of Cluster objects selected for growth
n  for Cluster, ParentCluster in Graph:
   |  if Cluster is ParentCluster:
   |  |  select Cluster for growth
   if Weighted growth: /* only smallers Clusters are selected                */
n  |  find smallest cluster size
n  |  find clusters with smallers size
   Complexity: O(n)
```

Algorithm 6: SelectGrowthClusters

```
   Data: Cluster
   Result: ParentCluster
   ParentCluster = ClusterIndex (Cluster)
n  while ParentCluster neq Cluster:
   |  Cluster = ParentCluster
   |  ParentCluster = ClusterIndex (Cluster)
   Complexity: O(n)
```

Algorithm 7: GetParentCluster

Bibliography

[1] A. Y. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of PHysics*, 303(1):2–30, January 2003.

[2] A. C. Doherty T.M. Stace, S. D. Barrett. Thresholds for topological codes in the presence of loss. *Physical Review Letters*, 102(20):200501, May 2009.

[3] N. H. Nickerson N. Delfosse. Almost-linear time decoding algorithm for topological codes. *Quantum Physics*, September 2017.

[4] S. D. Barrett T. M. Stace. Error correction and degeneracy in surface codes suffering loss. *Physical Review*, 81:022317, February 2010.

[5] G. Zemor N. Delfosse. Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel. *Quantum Physics*, March 2017.

[6] N. Nickerson. *Practical fault-tolerant quantum computing.* PhD thesis, Imperial College London, December 2015.

[7] D. Browne. Lectures on topological codes and quantum computation. chapter 1-5. bit.do/topological.