Quantum Error Correction

Correlated Decoding as Bilinear Programming

Ben Criger

October 27, 2016

1. Introduction

The toric code is a popular 2D

The assignment of an error to a given syndrome in the toric code can be reduced to minimum-weight perfect matching if X errors and Z errors occur independently, since the probability of an error chain can be expressed as a function of the distance between the syndromes. This argument changes if there's an independent source of Y errors, though $(p_Y \neq p_X p_Z)$, because the probability of an error that triggers an X syndrome is now correlated with the probability of an error that triggers a Z syndrome. Existing approaches for this problem [2, 1] rely on *reweighting*, adjusting the edge weights in an X syndrome graph based on where Z errors have been inferred, or vice versa. I have beef with this:

- It multiplies runtime by a factor of at least two, since the matching problem must be solved for one graph before the result can be applied to the other. This factor may be larger if we iterate, using each matching to reweight the other graph until convergence.
- Performance can, in principle, depend on which graph we match first. It may be the case that both orders (XZ and ZX) have to be attempted in parallel to get a decent solution. Even then, we have no guarantee that Y errors can be accurately inferred given a decoding procedure that puts X and Z steps in order like this.

We'd prefer to have a solution that can handle Y errors 'out of the box', to try to put them on the same footing as X and Z. In the rest of this document, I show how arbitrary-length Y-chains can be accurately weighed, provided that the problem is generalised to a bilinear program.

2. Weights

Bounding Box

3. Optimisation Problem

References

[1] N. Delfosse and J.-P. Tillich. A decoding algorithm for CSS codes using the X/Z correlations. *ArXiv e-prints*, January 2014.

[2] A. G. Fowler. Optimal complexity correction of correlated errors in the surface code. *ArXiv e-prints*, October 2013.

Appendices

A. Measuring a Different Code Every Round

If a bilinear program-based decoder doesn't work for whatever reason, do we have the option to measure a different set of stabilisers at every round that commute with the previous round's (e.g. CSS codes with Y and X strings instead of X and Z) and reconstruct the error history from there? A program of research suggests itself, where we go back to DKLP and figure out which errors form chains in 3D given the cycle of stabilisers that we measure.