Scalable decoders for quantum surface codes

by Sumeet Shirgure

USC, Fall 2022

1 Introduction

This article is intended to be a report of my term project for EE 599. Rather than a project report, this reads more like a letter, where I mostly describe recent developments and sometimes give my opinions on the matter. I understand that in my proposal I stated I will be studying and comparing two specific decoding algorithms for surface codes, but hopefully upon further reading it will be clear as to why I decided not go through with it. It turned out that both of them were based on related ideas, and would be combined into a single, better decoder.

Instead I will review some challenges in surface code decoding that present themselves when we might try to scale up the number of topologically encoded logical qubits, which was my original motivation to study said topics to begin with.

Section 2 briefly reviews the surface code, and motivates the decoding problem. Section 3 discusses the history and recent developments in decoding algorithms for the surface code. Section 4 discusses the the area of superconducting electronics and how it could be relevent when we try scaling up the surface code. Section 5 discusses alternatives to minimum weight matching based algorithms.

2 Surface codes

Surface codes like the toric and planar codes have become quite popular recently. A recent landmark being the experiment [1] on Google's Sycamore processor, where the objective was to remember one qubit of information encoded in a surface code over a period of time extending beyond any single constituent physical qubit's lifetime.

Surface codes were introduced in [7]. They are usually defined on a square lattice, with the stabilizer operators defined around each vertex $v : X_v \equiv \Pi_{q \in v} X_q$ and each face $f : Z_f \equiv \Pi_{q \in f} Z_q$. The stabilizer subspace of the code is also sometimes identified with the ground state of the Hamiltonian $H = -\sum_v X_v + \sum_f Z_f$.

Section IV of [7] discusses the physical notions behind error correction. We assume uncorrelated X and Z errors. Since this is a CSS code [19], we can focus on one of the X or Z errors, and use the same ideas for the other kind on a dual graph.

A chain of qubit errors on the lattice is only detected at its endpoints. Given multiple non-trivial syndrome measurements, the weight of chains that could have caused the syndrome corresponds to an increase in the energy of the system from its ground state.

Moreover the X and Z stabilizer errors have an interpretation of there being anyonic quasiparticle excitations present on the surface. These quasiparticles are of "charge" and "flux" types respectively with characteristic energies. The chains of errors are then analogous to electric and magnetic flux tubes. The ground state thus represents an anyonic vaccuum so to speak. This picture is quite insightful when it comes to the threshold for logical error supression (section 4).

If we take measurement errors into account, then repeated measurements add a time-like dimension to the lattice, the analysis of which is otherwise mostly the same.

That being said, minimizing the total energy of chains in space-time that show a syndrome is then reduced to finding a minimum weight matching on a weighted graph with weights that go as $\Delta_s \log\left(\frac{p}{1-p}\right) + \Delta_t \log\left(\frac{q}{1-q}\right)$, where Δ_s and Δ_t are space and time-like distances and p, q are data and measurement error rates respectively (refer to [7] for details.) It is straightforward to see that this is proportional to the log-likelihood of presence of such a chain. Logarithms are convenient because they convert multiplication of edge probabilities to a sum of log-likelihoods. The connection to energy comes from statistical physical considerations, which also gives the said accuracy threshold.

To quote [7] (section V), "we assume this classical computation [of finding the minimum weight matching] can be performed instantaneously and flawlessly". While not entirely incorrect, the latency with which we are able to do so indeed puts a lower bound on the gate times for the logical qubit. Furthermore, when the number of logical qubits grows, the challenge is to do it "instantaneously" *in parallel* while not requiring too much space, time, energy or entropy *per qubit*. This necessitates research into fast matching algorithms..

3 Decoding algorithms for surface codes

The initial approach to solving this matching problem was by constructing a complete graph with a weighted edge between every pair of non-trivial syndrome measurement (called "defects"), and using the famous[1] Blossom algorithm [8] on the resulting graph. While correct, it's not the most effecient, in that it has a worst case run time of $O(mn^2)$, where the input graph has m edges and n vertices, which is $O(n^4)$ for the kinds of graphs we provide.

The authors of [5] provide a seemingly different decoding algorithm which relies on *fusing* clusters of errors. Their idea is to start with an "island" for every non-trivial syndrome, "grow" each island uniformly until two or more fuse together into a single island, and keep repeating this for every island with an odd number of defects. Whenever two islands collide, there must be a path between *some* unmatched node in each of the two islands in the resulting matching.

1. The Blossom algorithm really is quite well known to anyone with background in theoretical computer science and graph theory. Personally I'm quite surprised by its relevance in quantum physical phenomena.

While the analysis in the paper considers both erasure errors and Pauli errors, this is roughly what is happing. Some researchers noticed the similar threshold values for this and the one obtained from solving the minimum weight matching, which raised suspicions that the faster algorithm really might be solving the latter exactly. I believe that is indeed the case.

To see why, let's consult this [9] paper by Fowler, which proves that minimum weight matching for such codes can be solved *exactly* for the kinds of graphs in topological quantum error correction (TQEC), with parallel computers whose numbers grow linearly with number of physical qubits, while taking $O(1)$ time. This itself should have been the first hint that an $O(n)$ time algorithm for solving it exactly must be possible, considering that we can just serialize this entirely.

[9] does this by casting the matching problem as a linear program. The radii of the "islands" in some norm are really the dual variables y_e in that same LP. Compare figure 4 in [9] with figure 3 in [5]. I believe the small discrepancy in thresholds in the latter can be explained by the fact that the so called Union-Find [5] decoder only grows the islands in integer amounts, but the complete linear programming formulation requires fractional variables. Of course there are other details surrounding constructing the matching itself when two islands merge, which is taken care of using *alternating trees* and the same ideas of from [8]

I found two implementations [12] ("pymatching" version 2) and [23] ("fusion-blossom") that claim linear time perfect matching for TQEC graphs. [23] also provides animations that show the ideas discussed above. They cite each other, and have yet to write papers on their software, but other than maybe a few code-level optimizations, I really don't see how they could be too different from being a serial implementation of the ideas from Fowler's original paper [9].

Note that these fast algorithms are only possible because the graphs on which we are solving the minimum weight matching have some special structure. Consider a matching M on a graph G that has two edges, e_1 between nodes a and b and e_2 between nodes c and d, and e_1 intersects e_2 at x. If there is a metric on this space, then from its corresponding triangle inequality on triangles a, c, x and b, d, x, we can see that it is always better to match either a with c or a with d.

It is a well known result in topology/graph theory [3] that planar graphs are sparse (i.e the edges grow linearly with the vertices $m \sim O(n)$). Moreover, recently it was also shown [2] that perfect matching for planar graphs is in class NC [2]. Perhaps this can be considered a generalisation of [9], although it's not as strong as needing just $O(n)$ processors and $O(1)$ time. This means that we would never explore more than a linear number of edges.

2. Class NC ("Nick's Class") is the set of decision problems solvable by $O(n^k)$ parallel processors in time $O(\log^c n)$ for problem instances growing as n. Just as the class P can be thought of as problems effeciently solvable on a standard computer, NC is the class of problems that are *effeciently parallelisable* on polynomially many computers. (Usually understood with a PRAM [14] model). Of course, $NC \subseteq P$ because we can simulate any algorithm in NC serially to get an algorithm in P.

The ideas presented in [9] are very interesting for other reasons as well. They enable research into what are called application specific integrated circuits (ASICs) for parallelised implementations of the decoding algorithm, as we will see in the following section.

4 Superconductivity

Decoders (ASICs or otherwise), when implemented on CMOS circuits are prone to high thermal emissions, which make them impractical to be of any use near the frigid environment of the quantum computer itself. The same is true for control electronics that are responsible for applying qubit gates.

If this apparatus is situated externally, there's likely some kind of cabling that crosses a thermal barrier. Presently, the plan to scale the surface code seems to rely on better cabling and microwave engineering to accomodate more qubits on a small chip. This can be expensive, both fiscally and in terms of latency.

A more effecient alternative might be the use of superconducting electronics for implementing these controllers and ASICs. Superconductivity can be considered a phase of matter characterized by a critical temperature, below which the material exhibits zero electrical resistance, and all magnetic fields are expelled, in what's called the Meissner effect.

Interestingly, the authors of [7] also give an interpretation of the accuracy threshold for toric codes as a similar kind of phase transition, where above a threshold physical error rate (which is analagous to temperature) the errors cause large thermal fluctuations destroying superconductivity.

This loss of electrical resistance allows very low power dissipation. Basically we only need power for gate switching and can transport information without drag. In particular, a loop of electric current ("supercurrent") will persist indefinitely with no power source. Moreover, such low temperature regimes are also necessary to bring about the quantum effect needed for qubits themselves to be realised. Superconducting qubits are essentially quantum (an-harmonic) inductance-capacitance oscillators [16] with a non-linear inductance generated by a Josephson junction. The quantum physics behind superconductivity and Josephson junctions also describe other useful phenomena like quantization of magnetic flux. These enable a class of logic families for classical computation specific to superconducting electronics that depend on these effects, like rapid single flux quantum (RSFQ) logic [18]. They promise immense improvements over conventional computing techonologies like CMOS, boasting picosecond level gate times, 100s of GhZ of operating frequency and a reduction of power consumption by orders of magnitudes.

Various organizations in the US government have invested in research involving superconducting computers since as early as 1950s. I'm guessing the high cost of refridgeration deterred their practical viability early on. However the recent boom of quantum computing, coupled with the end of Moore's law has probably given it a much larger incentive.

Currently IARPA[3] is leading the research into superconducting and low power computing. I myself had the opportunity to contribute to one of their programs while working under Prof. Pedram here at USC. The "Supertools" program developed toolchains for automated design of superconducting circuits.

Such EDA tools have been fine tuned and perfected for CMOS over decades. But we're still in the early stages of designing even basic superconducting chips like ALUs and adders, much less automating fabrication of one with enough bits.[4] So implementing surface code decoder ICs is pretty far from realistic as of now. However, that hasn't stopped research in similar directions [13], [15], [21].

5 Beyond minimum weight matchings

While minimizing weights of error chains is an excellent strategy that is effeciently doable (and even parallelizable at least in theory) it may not be the optimal one. For one it performs poorly in presence of correlated Y errors arising from depolarizing noise. Secondly, the objective of a decoder is to ultimately minimize the *logical errors* it makes in guessing the logical state after error detection. The optimal strategy should maximize this conditional probability of the predicted logical state given the observed syndrome, aptly called *maximum likelihood decoding* [7].

Several efficient algorithms for MLD have been proposed under varying noise models; for example [4] (linear time decoding of erasure errors, which is simlar to the fusion algorithm from 3), and [6] ("tensor network" decoding) that have a theoretical threshold higher than that given by matching decoder. The report in [1] suggests that tensor network decoding is impractical when scaling up the surface code. Instead it relies on a modified min-weight matching decoder [11] with weights obtained with belief propagation [20] to account for "hyperedge faults" which are used to model more than one syndrome bit being triggered from multiple error simultaneous data errors.

Maximum likelihood decoders rely on estimating the parameters of a probability distribution $\mathbb{P}_\theta$. Recent advances in machine learning and artificial intelligence have shown probability distributions to be very effectively learnable using a variety of methods like naive Bayesian inference, Boltzmann machines and neural networks.

3. IARPA does some pretty amazing stuff. https://www.iarpa.gov/research-programs?office_name=collection. You might also be interested in their upcoming LogiQ and ELQ programs that aim to create and entangle error corrected logical qubits.

4. One might even say the problem is Superdiffficult™

Deep neural networks are by nature highly parallelisable and have proved to be extremely effective in learning all kinds of probability distributions just from its samples. Moreover there already exists abundant infrastructure like GPUs and Google's tensor processing units that are tailor made for exactly such computational models. So, despite of the immense hype and saturation surrounding the field that I'm not a big fan of either[5], it would be highly convenient if we're able to use it for the decoding problem at scale. Of course, tensor computation being on the other end of the thermal spectrum can only be used in an off-chip setting.

There already has been some work in this regard [17], [10], [22]. The authors in [17] simply construct a linear neural network mapping syndrome bits to predicted noise and train the network to minimize the cross entropy loss between the predicted distribution and samples generated from simulated circuit level noise. Note that this approach works for any CSS code given its parity matrix. Meanwhile later works try to come up with more sophisticated networks like those with convolutional layers to decode surface codes. [17] also states that they trained a new network for different values of p (physical error rates) which is somewhat of a drawback. Ideally it should be a single map that is robust to variations in error rates. But the improvement in threshold over min-weight matching was substantial.

One thing I noticed while reading them is that none of the above considered measurement errors, i.e. they assume 2d instead of (2+1)d syndrome measurements. So I also tried extending the idea in [17] to work with time-like errors (by using code from [12] and some custom code) but it's quite difficult to get it working[6].

Bibliography

[1] Rajeev Acharya, Igor Aleiner, Richard Allen, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Juan Atalaya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Joao Basso, Andreas Bengtsson, Sergio Boixo, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Michael Broughton, Bob B. Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Josh Cogan, Roberto Collins, Paul Conner, William Courtney, Alexander L. Crook, Ben Curtin, Dripto M. Debroy, Alexander Del Toro Barba, Sean Demura, Andrew Dunsworth, Daniel Eppens, Catherine Erickson, Lara Faoro, Edward Farhi, Reza Fatemi, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, William Giang, Craig Gidney, Dar Gilboa, Marissa Giustina, Alejandro Grajales Dau, Jonathan A. Gross, Steve Habegger, Michael C. Hamilton, Matthew P. Harrigan, Sean D. Harrington, Oscar Higgott, Jeremy Hilton, Markus Hoffmann, Sabrina Hong, Trent Huang, Ashley Huff, William J. Huggins, Lev B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Pavol Juhas, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Tanuj Khattar, Mostafa Khezri, Mária Kieferová, Seon Kim, Alexei Kitaev, Paul V. Klimov, Andrey R. Klots, Alexander N. Korotkov, Fedor Kostritsa, John Mark Kreikebaum, David Landhuis, Pavel Laptev, Kim-Ming Lau, Lily Laws, Joonho Lee, Kenny Lee, Brian J. Lester, Alexander Lill, Wayne Liu, Aditya Locharla, Erik Lucero, Fionn D. Malone, Jeffrey Marshall, Orion Martin, Jarrod R. McClean, Trevor Mccourt, Matt McEwen, Anthony Megrant, Bernardo Meurer

5. *Believe me.*

6. At the time of writing I'm still trying. I realise the semester is over, but if I'll still work on it over the holidays if possible.

Costa, Xiao Mi, Kevin C. Miao, Masoud Mohseni, Shirin Montazeri, Alexis Morvan, Emily Mount, Wojciech Mruczkiewicz, Ofer Naaman, Matthew Neeley, Charles Neill, Ani Nersisyan, Hartmut Neven, Michael Newman, Jiun How Ng, Anthony Nguyen, Murray Nguyen, Murphy Yuezhen Niu, Thomas E. O'Brien, Alex Opremcak, John Platt, Andre Petukhov, Rebecca Potter, Leonid P. Pryadko, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Negar Saei, Daniel Sank, Kannan Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, Christopher Schuster, Michael J. Shearn, Aaron Shorter, Vladimir Shvarts, Jindra Skruzny, Vadim Smelyanskiy, W. Clarke Smith, George Sterling, Doug Strain, Marco Szalay, Alfredo Torres, Guifre Vidal, Benjamin Villalonga, Catherine Vollgraff Heidweiller, Theodore White, Cheng Xing, Z. Jamie Yao, Ping Yeh, Juhwan Yoo, Grayson Young, Adam Zalcman, Yaxing Zhang, and Ningfeng Zhu. Suppressing quantum errors by scaling a surface code logical qubit. 2022.

[2] Nima Anari and Vijay V. Vazirani. Planar graph perfect matching is in nc. 2017.

[3] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Elsevier, New York, 1976.

[4] Sergey Bravyi, Martin Suchara, and Alexander Vargo. Efficient algorithms for maximum likelihood decoding in the surface code. *Physical Review A*, 90(3), sep 2014.

[5] Nicolas Delfosse and Naomi H. Nickerson. Almost-linear time decoding algorithm for topological codes. *Quantum*, 5:595, dec 2021.

[6] Nicolas Delfosse and Gilles Zémor. Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel. *Physical Review Research*, 2(3), jul 2020.

[7] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002.

[8] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[9] Austin G. Fowler. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time. 2013.

[10] Spiro Gicev, Lloyd C. L. Hollenberg, and Muhammad Usman. A scalable and fast artificial neural network syndrome decoder for surface codes. 2021.

[11] Oscar Higgott, Thomas C. Bohdanowicz, Aleksander Kubica, Steven T. Flammia, and Earl T. Campbell. Fragile boundaries of tailored surface codes and improved decoding of circuit-level noise. 2022.

[12] Oscar Higgott and Craig Gidney. Pymatching v2. https://github.com/oscarhiggott/PyMatching, 2022.

[13] Adam Holmes, Mohammad Reza Jokar, Ghasem Pasandi, Yongshan Ding, Massoud Pedram, and Frederic T. Chong. Nisq+: boosting quantum computing power by approximating quantum error correction. 2020.

[14] Joseph JáJá. *An Introduction to Parallel Algorithms*. Addison Wesley Longman Publishing Co., Inc., USA, 1992.

[15] Mohammad Reza Jokar, Richard Rines, Ghasem Pasandi, Haolin Cong, Adam Holmes, Yunong Shi, Massoud Pedram, and Frederic T. Chong. Digiq: a scalable digital controller for quantum computers using sfq logic. 2022.

[16] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. Superconducting qubits: current state of play. *Annual Review of Condensed Matter Physics*, 11(1):369–395, 2020.

[17] Stefan Krastanov and Liang Jiang. Deep neural network probabilistic decoder for stabilizer codes. *Scientific Reports*, 7(1), sep 2017.

[18] K.K. Likharev and V.K. Semenov. Rsfq logic/memory family: a new josephson-junction technology for sub-terahertz-clock-frequency digital systems. *IEEE Transactions on Applied Superconductivity*, 1(1):3–28, 1991.

[19] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, England, dec 2010.

[20] Judea Pearl. *Probabilistic reasoning in intelligent systems*, volume 88. Elsevier, 2014.

[21] Gokul Subramanian Ravi, Jonathan M. Baker, Arash Fayyazi, Sophia Fuhui Lin, Ali Javadi-Abhari, Massoud Pedram, and Frederic T. Chong. Better than worst-case decoding for quantum error correction. 2022.

[22] Giacomo Torlai and Roger G. Melko. Neural decoder for topological codes. *Physical Review Letters*, 119(3), jul 2017.

[23] Yue Wu. Fusion blossom. https://https://github.com/yuewuo/fusion-blossom, 2022.