Vol.2022-QS-5 No.27
2022/3/25

エラー確率が不均質な表面符号の復号のフェニック木を用いた高速化

新居 智将[1,2,a)]　鈴木 泰成[2,3]　徳永 裕己[2]

概要：表面符号は現在量子誤り訂正符号として最も注目されているものの 1 つである．表面符号の復号方法の 1 つに，エラーの推定を最小重み完全マッチング問題に帰着して復号する方法がある．量子ビットによってエラーが起きている確率が均一でない場合，重みが不均質な最小重み完全マッチング問題を解くことで復号の精度が向上することが知られている．しかし，その際に必要な計算量は確率が均一である場合に比べて大きくなってしまう．この発表では，一定の仮定の下でフェニック木を用いることで，最小重み完全マッチング問題による表面符号の復号で必要となる計算量を削減する方法を提案する．

Fast decoding algorithms with Fenwick trees for surface codes under non-uniform errors

Abstract: Surface codes are one of the most promising quantum error-correcting codes. The estimation task of recovery operations for surface codes can be reduced to an instance of a minimum-weight perfect matching problem. When the error probabilities of physical qubits are not uniform, solving a minimum-weight perfect matching problem with non-uniform weights is required to improve the performance of error correction. However, the decoding algorithm under non-uniform weights requires longer than the uniform cases. In this paper, we propose a fast decoding algorithm for surface codes that uses a Fenwick tree as a key component. Our algorithm can reduce the time complexity with an approximation and achieve lower logical error rates than existing methods.

1. 概要

従来の計算機よりも計算量理論的に高速に計算を行う枠組みとして，物質の量子的な性質を用いた量子計算機が注目を集めている [1]．量子計算機は従来の計算機よりもエラーが大きいため，素朴に拡大しても意味のある計算を行うことが出来ない．量子計算機のエラーを効率的に削減する枠組みで最も有望な手法の一つが量子誤り訂正符号によって誤りを訂正しながら計算を行う誤り耐性量子計算である [1], [2]．誤り耐性量子計算ではエラーが蓄積するよりも十分早くエラー検出のためのスタビライザー測定と呼ばれるパリティ検査を行い，得られたシンドローム値から計算に追従してエラーを推定し訂正することができる．特に，表面符号 [3], [4] は二次元的に配列された量子ビットで容易に実装でき，高い誤り訂正の性能を誇ることから実現が最も期待されている [5], [6].

表面符号を用いた誤り耐性量子計算の実現で鍵となる課題の一つが, 計算中に生じるエラーをシンドローム値から高速に推定するための復号アルゴリズムの高速化である．表面符号においてシンドローム値から適切なエラー訂正操作を推定する手続きは，３次元格子上における最小重み完全マッチング (Minimum-weight perfect matching, MWPM) 問題に帰着されることが知られている [5], [7]．計算中はスタビライザー測定の速度よりも高速に MWPM を解き続ける必要があるため，表面符号の誤り訂正能力を高めるためには，より高速に MWPM を解くアルゴリズムが必要となる．このため，これまで MWPM を高速に解くために様々な手法と実装が提案されてきた [8], [9], [10], [11], [12], [13]. この MWPM のインスタンスを生成する過程で，重みづけされた３次元格子上の様々な二点間の最短距離を計算するサブルーチンが必要となる．もし，各量子ビットに生じるエラーが一様であれば，３次元格子上の辺の重みは一様となる．この時，二点間の距離はマンハッタン距離を定

[1] 東京大学 理科学部 物理学科
[2] NTT コンピュータ&データサイエンス研究所
[3] JST さきがけ
[a)] tom-arai-1307@g.ecc.u-tokyo.ac.jp

数時間で計算し容易に求めることが出来る．しかし，量子ビットに生じるエラー確率が不均質である場合，格子上の辺の重みもまた不均質となる．この時に２点間の最短距離を求めるには素朴にはダイクストラ法やワーシャルフロイド法などのグラフアルゴリズム [14] が必要となり，その計算量は３次元格子のノード数に対して多項式的に増加する．しかし，ダイクストラ法やワーシャルフロイド法などのアルゴリズムを用いるアプローチはリアルタイムに実行するには遅すぎるため，大きな符号で高速にエラーの推定を行う実装ではエラー確率が不均質である場合も均質なエラーを仮定して復号を行うアプローチが一般的であった [10], [11], [12], [13].

この研究ではフェニック木のデータ構造 [15] を用いることで，２点間の最短距離を近似的に求める従来より計算量的に高速なアルゴリズムを提案する．C++で実装したベンチマークによれば，本手法により，実用化で必要とされる符号距離 21 などで 180 倍ものの高速化が得られている．さらに，不均質さを無視して一様な重みづけを仮定した MWPM を解く場合に誤り訂正のしきい値が 6%となる状況で，本手法は誤り訂正のしきい値 9%を実現している．本手法は従来の復号アルゴリズムにおいてボトルネックの一つであり性能劣化の主要因であった不均質なエラーを計算量と実速度の両方で高速に扱う手法を提案するものである．これにより，従来より高速な表面符号のエラー推定が実装可能となるため，より巨大な誤り耐性量子計算機の実装が可能となる．

2. 背景

2.1 表面符号

表面符号 [3], [4], [5] は二次元平面上に並べられた量子ビットで実装が可能で，高い精度でのエラー訂正が可能なため，超伝導量子ビットなどをはじめとした集積化可能な量子ビットで実現が可能な量子誤り訂正符号として注目を集めている．表面符号では**図 1** の左図に示すように，二次元格子に並んだ量子ビットを用いて一つの論理量子ビットを表現する方式である．左の図において，黒い丸はデータを保持する量子ビットを表し，赤または青いタイルは，頂点の場所に配置されている量子ビットのパウリ X および Z 測定を行うことを意味している．このパウリ測定はスタビライザー測定と呼ばれ，得られた値はシンドローム値と呼ばれる．量子ビットに対して確率的にパウリ操作のエラーが生じるようなエラーモデルを考えた場合，パウリ $X(Z)$ のシンドローム値を得る操作は，観測対象となる量子ビットに乗っているパウリ $Z(X)$ エラーの個数の偶奇を知ることに相当する．表面符号の符号距離 d は格子の幅および高さで定義され，表面符号は高々 $d-1$ 量子ビットまでに作用する任意のビット反転や位相反転のエラーを検出することができる．

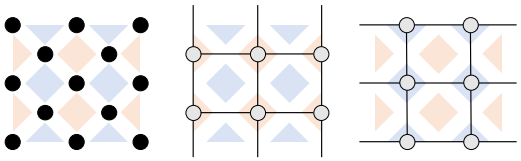


図 1: 表面符号の例．左の図は表面符号のデータ量子ビットの配置を表す．中央と右の図は，そこから得られる X および Z のスタビライザー測定値の場所割り当てと，復号の対象となる正方格子のグラフを表す．

論理ビットに生じているエラーを取り除き元の論理状態に回復するためのパウリ操作の推定は以下のように求めることができる．図 1 の中央および右図のように，パウリ $X(Z)$ スタビライザー測定の結果をノードとし（灰色の丸），データを保持する量子ビットをエッジとしたグラフを考える．ここで，エッジに対応する量子ビットにパウリ $Z(X)$ エラーが生じると，エッジが繋ぐ二つのノード（境界の場合は一つのノード）のシンドローム値の偶奇が反転される．最小距離復号と呼ばれる手法の復号の考え方では，観測されたノードの偶奇を再現する最も高確率で生じる辺の集合を求め，選ばれた辺に対応するパウリ操作を行うことでエラーを回復することができる．

上記の推定は以下のように定義された全結合グラフの最小重み完全マッチング問題 (MWPM) に帰着することができる．グラフの頂点は，図 1 の中央および右の正方格子のグラフにおいてパリティが奇となっているノードである（パリティが奇のノードが奇数個の場合は，境界に相当するノードを一つ加えて偶数とする）．ノード間の辺の重みは，図 1 の正方格子で各辺の重みを $-\log(p/(1-p))$（ただし p は対応する量子ビットにエラーが生じる確率）として２点間の最小重み経路の重みで定義する．こうして作られたグラフの最小重み完全マッチングが分かれば，そこから最小距離復号に相当する回復操作を計算することができる．MWPM の問題はエドモンドの花アルゴリズム [16] で多項式時間で解けるほか，より高速な近似アルゴリズムがいくつか知られている [9], [10], [11].

MWPM のインスタンスを生成するうえで，全てのパリティが奇数となっているノード間の最小距離を求める必要がある．この重みは，辺のコストが一様であれば図 1 の座標におけるマンハッタン距離として容易に計算することができる．一方，辺の重みが一様でない場合で，かつ，エラーレートが十分に小さい領域では，奇パリティのノード数の減少に伴い小さくなり MWPM の問題サイズも小さくなるものの，最小重みの経路を探索する問題サイズは小さくなるとは限らないため，MWPM のインスタンス生成に MWPM を解く時間よりも長い時間が必要となりうる．従って，不均質な２次元格子で与えられた２点間の重みを高速に求める手法は，高速で正確な復号を行う上で重要で

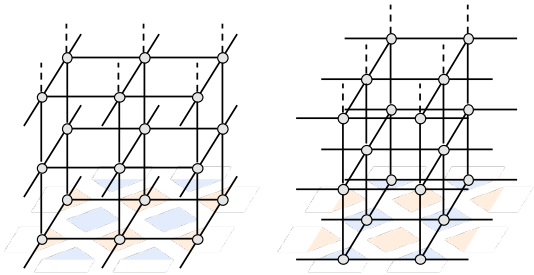


図 2: 測定にエラーがある場合に復号の対象となる 3 次元格子

ある．

実際の誤り訂正では，パリティ測定の操作自体にエラーが低確率で生じる．このような場合は，**図 2** に示すような 3 次元の格子を構成することで復号が可能となる．ここで高さ方向の辺は測定エラーの有無に対応し，その重みは測定エラーの確率より同様の式で定まる．この時，任意の頂点は少なくとも $\lfloor d/2 \rfloor$ の長さで境界にマッチできるため，典型的には高さ方向が d となる立方体を対象として復号問題を解くことが多い．復号問題の MWPM への帰着に関する詳細については文献 [5] および [17] を参照されたい．

2.2 フェニック木

フェニック木 (Fenwick tree) [15] には様々な操作を載せることができるが，本手法で用いる 2 次元フェニック木は $N \times N$ の 2 次元配列 $F[i][j]$ $(1 \leq i, j \leq N)$ に対して次の操作をともに $O(\log^2 N)$ で行う事が可能なデータ構造である．

- chmax(l_1, l_2, x) $l_1 \leq i \leq N$, $l_2 \leq j \leq N$ をみたす (i, j) について，$F[i][j]$ の値を $\max(F[i][j], x)$ に変更する．
- get(a, b) $F[a][b]$ の値を取得する．

先にまず，1 次元の場合について述べる．これは，長さ N の 1 次元配列 $\tilde{F}[i]$ に対して次の操作をともに $O(\log N)$ で行う事が可能なデータ構造である．

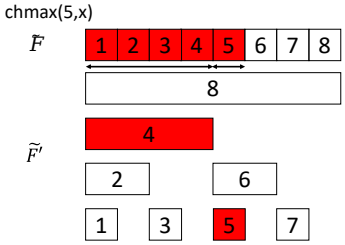
- chmax(r, x) $1 \leq i \leq r$ をみたす i について，$\tilde{F}[i]$ の値を $\max(\tilde{F}[i], x)$ に変更する．
- get(a) $\tilde{F}[a]$ の値を取得する．

ただし，配列は最初は全て 0 で初期化されているものとする．

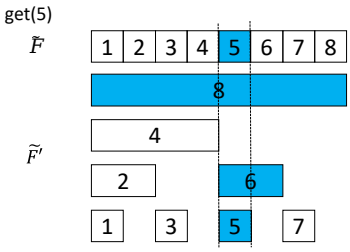
これには次の事実を利用する．

正整数 k に対して，$f(k)$ を 2^i が k を割り切る最大の i に対する 2^i の値で定める．このとき，

- 任意の正整数 k に対して，$1 \leq m \leq \lfloor \log_2 k \rfloor + 1$ なる整数 m と正整数 $i_1 < i_2 < \cdots < i_m$ が存在して，半開区間 $(0, k]$ は $(i_t - f(i_t), i_t]$ によって重なりなく覆われる．
- 任意の正整数 k に対して，$k \in (i - f(i), i]$ であるような N 以下の正整数 i は高々 $\lfloor \log_2 N \rfloor + 1$ 個である．



(a) chmax 操作，赤色が更新される要素



(b) get 操作，青色が取得に用いる要素

図 3: フェニック木の操作

これを用いて実際に管理する長さ N の配列 $\tilde{F}'$ には区間 $(i - f(i), i]$ に行われた max 操作のうち値が最大であったものを保管する．すなわち，

- chmax(r, x) $(0, r]$ を覆う高々 $\lfloor \log_2 N \rfloor + 1$ 個の区間 $(i - f(i), i]$ に分割し，$\tilde{F}'$ において対応する配列の要素それぞれについて max を取る操作を行う．
- get(a) a を含む高々 $\lfloor \log_2 k \rfloor + 1$ 個の区間 $(i - f(i), i]1$ 全てについての max を取る．

この時それぞれの操作は $O(\log N)$ で行う事ができ，またすべての更新操作は各点を含む区間のうちちょうど 1 つに反映されているから，正しく機能する．1 次元フェニック木 $(N = 8)$ 操作の例を**図 3** 以下に載せる．

また，区間をすべて反転させることで，

- chmax(l, x) $l \leq i \leq N$ をみたす i について，$\tilde{F}[i]$ の値を $\max(\tilde{F}[i], x)$ に変更する．
- get(a) $\tilde{F}[a]$ の値を取得する．

の 2 種類の操作に対応するフェニック木も同様に考えることができる．

2 次元のフェニック木については区間を反転させた後の 1 次元のフェニック木の $\tilde{F}'$ において管理していた区間を $I_1, \ldots, I_N$ とするとき，$F'[a][b]$ においては $I_a \times I_b$ の矩形

領域に対して行われた操作において保存するようにすれば良い．このとき，$[l_1, N+1) \times [l_2, N+1)$ は $[l_1, N+1)$ と $[l_2, N+1)$ のそれぞれの区間への分割の直積を考えることで，高々 $\{\lfloor \log_2 N \rfloor + 1\}^2$ 個の矩形領域に分割され，同様に $F[a][b]$ はそれぞれ高々 $\{\lfloor \log_2 N \rfloor + 1\}^2$ 個の矩形領域 $I_a \times I_b$ に含まれる事が分かる．よって，このようなデータ構造の下で最初に述べた 2 つの操作を $O(\log^2 N)$ で行う事が出来る．2 次元フェニック木 ($N = 4$) における chmax の操作の例を**図 4** に載せる．

3. 手法

3.1 提案手法の概要

この研究では表面符号の復号で必要となる $d \times d \times d$ の三次元格子上の 2 点間の最小コストを，以下のようなノイズモデルの仮定の下で，フェニック木を用いて近似的に高速に計算するアルゴリズムを提案する．本手法ではノイズの前提として，ほとんどの量子ビットは均一的な性能を持ち，一部の量子ビットが何らかの理由でエラーを起こしやすくなっているような場合を考える．このような状況は，集積化の地点でエラーが大きな量子ビットが生じている状況 [18]，宇宙線などの衝突によって一部の量子ビットのエラーが一時的に上昇している状況 [19]，あるいは，測定の信号が消失したり信頼性が低いなどでシンドロームの値が信頼できない状況 [20] などで生じる．このとき，格子上ではほとんどの辺が均一的なコストを持ち，一部の辺だけがコストが小さくなっている状態になる．

本手法による計算量の改善は表 1 に纏められている．ここで N はパリティが奇数となっている格子点の数であり，M はコストが小さくなっている辺の本数である．我々の目的は N 個のノードに対して全点対全点の最小距離を求めることである．ここで，N, M の値は d^3 に比例して大きくなる値ではあるが，エラー率が十分小さく，エラーが起こすやすくなるような量子ビットも多くないとみなせる実用的な状況では，$N, M \ll d^3$ とみなすことができる．重みづけされた 3 次元格子上の様々な二点間の最短距離を求める方法として最も素朴なベースラインとなる手法は，全格子点 (頂点数 $\sim d^3$) を用いたワーシャルフロイド (WF) 法であり，その計算量は $O(d^9)$ である．一方，重みが不均質であることを無視し，全てを均質だとして計算する場合は 2 点間の最小距離はマンハッタン距離として定数時間で計算できるため $O(N^2)$ となる．この手法は高速だが，不均質さを無視することによって後で述べるように誤り訂正の性能は大きく悪化する．

WF 法を改善する素朴な方法として，N 頂点についてダイクストラ法を適用するアプローチを考えることもできる．この手法ではある頂点を起点としたダイクストラ法の実施に $O(d^3 \log d)$ 必要とすると，全体のコストは $O(Nd^3 \log d)$ となる．この手法は WF 法よりは高速だが，そのコストは奇数でないパリティのノードを含む 3 次元格子の全頂点数 d^3 にあらわに依存してしまう．$N, M \ll d^3$ が成り立つ領域でのさらなる改善として，以下のような手法を考えることができる．ある 2 点間の経路は，均一な辺を何本か (0 本も含む) 通る区間と，エラー率が高い辺 (重みが低い辺) を通る区間を繰り返す．ここで，経路上において，コストが変化している辺から次のそのような辺までの間ではコストが均一な辺しか通らないため，この区間の最小コストはつねにマンハッタン距離に均一的な定数係数 C をかけたものとなる．よって，2 点対間の距離を求めなければならない格子点の集合とコストが変化している辺の端点となっているような格子点をあわせたものについて，格子点同士のコストをマンハッタン距離に C をかけたものとしてとして計算し，その上でコストが変化した辺の情報を反映させたグラフの上で WF 法を行っても同様に厳密な最短距離が得られる．このとき 2 点対間の距離を求めなければならない格子点の集合の大きさを N，コストが変化した辺の数を M として，計算量は $O((N+M)^3)$ となる．以下，この方法で最短経路を計算する手法を WF beta 法と呼ぶことにする．

WF beta 法は WF 法よりは近似なしに高速な計算が可能だが，物理量子ビットのエラー率はそのままに符号距離 d が増加すると，オーダーとしては $O(d^9)$ の計算量となってしまうため，さらなる改善が期待される．そこで，以下のような近似を導入しさらに効率的なアルゴリズムを提案する．我々は，重みづけされた 3 次元格子上で 2 頂点間を結ぶ経路の最小コストの経路は，重みを均質とみなした際の最短経路のいずれかと一致するとみなす．境界でない二つの頂点間の最小コストの経路を考えると，重みが均質とみなした経路のうち，最短でない経路のうち最も短い経路は，最短の経路に比べ 3 つ以上の辺を余分に通過する必要がある．従って，多くの辺のコストが一斉に変化しない限り，このような経路がよりコストが小さくなる可能性は低いと期待され，この近似により復号が誤ったものになる確率はさらに小さいと期待される．従って，格子上で最短経路を達成する経路のうちで重みの総和を最小化する経路のコストが分かれば，高い精度で 2 点間の経路の最小コストが求められる事が期待できる．この場合経路の単調性が保証されるため，この事実を用いて計算量を落とすことができる．なお，厳密には 1 本までは最短経路より多く通る経路を含めても単調性が保証されるため，本研究で提案するアルゴリズムはそのようなものも含めて最小値を計算している．詳細は以下に述べるが，その計算量は $O(N(N+M)\log d) \sim O(d^6 \log^2 d)$ となる．なお，このような近似は，すべての辺の重みが異なっているが，平均的な各辺のコストに対して，辺によるコストの差が小さいような場合についても同様に有効に成り立つと考えられる．

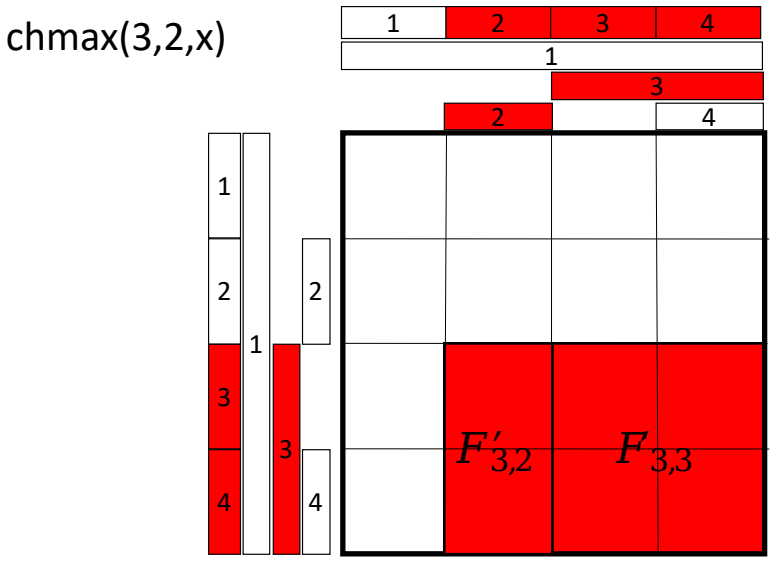


図 4: chmax 操作，マス目内の赤色領域 (2 つ) が更新される要素

	時間計算量	d への依存性	偶パリティのノード数に非依存	重みを考慮
WF	$O(d^9)$	$O(d^9)$		○
WF beta	$O((N+M)^3)$	$O(d^9)$	○	○
Dijkstra	$O(Nd^3 \log d)$	$O(d^6 \log d)$		○
Fenwick	$O(N(N+M)\log^2 d)$	$O(d^6 \log^2 d)$	○	部分的
Manhattan	$O(N^2)$	$O(d^6)$	○	

表 1: 時間計算量

3.2 提案手法の詳細

この手法では，3 次元格子上の 2 点間を結ぶ経路のうち，2 点のマンハッタン距離から高々 1 本までしか多く辺を通らないような経路の中に最小コストの経路が存在すると仮定し，最小コストを求めることで高速化を実現する．

シンドローム測定の結果はエラーの出た位置を表す格子点の集合 $V = \{(i_a, j_a, k_a)\}$ として，量子ビットの重みを表す情報は量子ビットの位置を表す隣接する格子点の順序付けされていない組および正常状態と比較したときのコストの減少幅を表す数値，すなわち $E = \{((i_b, j_b, k_b), (i'_b, j'_b, k'_b), c_b)\}$ によって与えられるとする．以下，V の要素を頂点と呼ぶ．

次の 2 つを計算する必要がある．

(a) 各頂点から境界までの最小コスト

(b) 2 頂点間の最小コスト

頂点 $u = (i_u, j_u, k_u)$ を 1 つ固定する．その頂点から次の位置関係にある頂点までの最短経路の中での最小コスト，および，余計な辺を 1 本までしか使わないような境界までの経路の中での境界までの最小コストを求める．

(i) $i_u \leq i_v,\ j_u \leq j_v,\ k_u \leq k_v$

(ii) $i_u \leq i_v,\ j_u \geq j_v,\ k_u \geq k_v$

(iii) $i_u \geq i_v,\ j_u \leq j_v,\ k_u \geq k_v$

(iv) $i_u \geq i_v,\ j_u \geq j_v,\ k_u \leq k_v$

(i) のパターンについて求められれば，適切な座標の反転によって他の 3 パターンについても求められるため，以下では (i) のパターンについて求める方法について述べる．

クエリの列 Q および 2 次元 Fenick 木 F を用意する．クエリの列は，次の要素を辞書順にソートしたものを用意する．これは頂点によらない．

- $\{(i_v, j_v, k_v, v, 0) | (i_v, j_v, k_v) \in V\}$
- $\{(i_e + 0.5, -j_e, -k_e, -1, c_e) | ((i_e, j_e, k_e), (i_e + 1, j_e, k_e), c_e) \in E\}$
- $\{(i_e, j_e + 0.5, k_e, -1, c_e) | ((i_e, j_e, k_e), (i_e, j_e + 1, k_e), c_e) \in E\}$
- $\{(i_e, j_e, k_e + 0.5, -1, c_e) | ((i_e, j_e, k_e), (i_e, j_e, k_e + 1), c_e) \in E\}$

また，2 次元 Fenick 木の要素はすべて 0 に初期化しておく．

次のようにクエリを処理する．$Q_q = (i, j, k, v, c)$ であったとする．

- $i < i_u, j < j_u, k < k_u$ のいずれかが成り立つときは，何も行わない．
- 上記でなく，$v \geq 0$ のとき

  頂点 u と頂点 v 間の境界を経由しない最短経路の最小コストとして，$C(|i-i_u|+|j-j_u|+|k-k_u|)-\mathrm{get}(j,k)$ を記録する．
- 上記でなく，$v < 0$ のとき

  $x = \mathrm{get}(\lfloor|j|\rfloor, \lfloor|k|\rfloor)$ として，更新操作 $\mathrm{chmax}(\lceil|j|\rceil, \lceil|k|\rceil, x+c)$ を行う．

ただし、$\lfloor x \rfloor$, $\lceil x \rceil$ はそれぞれ底関数と天井関数である。最後にすべてのクエリを処理した後の $|d-i_u|-\mathrm{get}(j_u,k_u)$, $|d-i_u|+1-\mathrm{get}(j_u+1,k_u)$, $|d-i_u|+1-\mathrm{get}(j_u,k_u+1)$ の値を保存しておく．これは，それぞれ 頂点 u から (d,j_u,k_u), (d,j_u+1,k_u), (d,j_u,k_u+1) までの最短経路の中での最小コストであり，頂点から境界までの最小コストを求める時に用いる．

3.2.1 (a) 各頂点から境界までの最小コスト

頂点 (i_u,j_u,k_u) から境界までの経路のうち余計な辺を 1 本までしか使わないようなものは (i, j_u+j', k_u+k') $(i \in \{0,d\}, |j'|+|k'| \leq 1)$ までの最短経路それぞれについて最小コストを求め，その最小値をとればよい．ここで，上の計算の中で

(i) (d,j_u,k_u), (d,j_u+1,k_u), (d,j_u,k_u+1)

(ii) (d,j_u,k_u), (d,j_u-1,k_u), (d,j_u,k_u-1)

(iii) $(0,j_u,k_u)$, $(0,j_u+1,k_u)$, $(0,j_u,k_u-1)$

(iv) $(0,j_u,k_u)$, $(0,j_u-1,k_u)$, $(0,j_u,k_u+1)$

への最短距離が求まっており，上の集合の要素を網羅していることから，これらから求めることができる．

3.2.2 (b) 2 頂点間の最小コスト

2 頂点 u,v 間の最小コストは境界を経由する経路と経由しない経路それぞれの中での最小コストのうち小さい方として求まる．経由するものは (a) で求めた各頂点から境界への最小コストの和として求まる．経由しないものについては $(i_v-i_u)(j_v-j_u)(k_v-k_u) \geq 0$ ならば頂点 u から頂点 v への最小コストとして，$(i_v-i_u)(j_v-j_u)(k_v-k_u) < 0$ ならば頂点 v から頂点 u への最小コストとして求まっているからこれを境界を経由するものと比較すればよい．

これにより，求めたいものが全て求まった．

3.3 提案手法の証明

上の手法の中で，2 頂点またはある頂点と境界上の点の間について，その 2 点間の最短経路の中で最小コストが求まっていたとする．境界を経由しない経路については，最短距離でないものは必ず最短経路から 2 本以上多く辺を通る事から，今回の近似の下では考えなくて良い．境界を経由するものについては，境界までの最短距離より 1 本まで多くの辺を通ることができるが，境界へ向かう向きと反対の方向に進むとやはり 2 本以上必要になる事から，それは境界へ向かう向きと垂直な方向である必要がある．よって，そのような経路は頂点からまっすぐ境界面に向かったときの終着点からマンハッタン距離が 1 以下である点への最短経路となっている．よって，上のアルゴリズムによって，2 点間のマンハッタン距離だけ辺を通る経路の中で最小コストが求まっていることを示せば良い．

2 頂点 u,v の座標を (i_u,j_u,k_u), (i_v,j_v,k_v) とし，対称性から $i_u \leq i_v$ かつ $j_u \leq j_v$ かつ $k_u \leq k_v$ のときについて求まっていることを示せば良い．以下，辺の情報 $((i_e,j_e,k_e),(i'_e,j'_e,k'_e),c_e) \in E$ を $\left(\frac{i_e+i'_e}{2}, \frac{j_e+j'_e}{2}, \frac{k_e+k'_e}{2}, c_e\right)$ に対応させたものからなる集合 E' を考える．なお，$(i_e,j_e,k_e,c_e) \in E'$ から，無向辺の情報は $((\lfloor i_e \rfloor, \lfloor j_e \rfloor, \lfloor k_e \rfloor), (\lceil i_e \rceil, \lceil j_e \rceil, \lceil k_e \rceil), c_e)$ と復元できる．頂点 u から頂点 v に向かう最短経路上では各座標を増加する方向にしか移動できない事から，頂点 u から頂点 v に向かう最短経路であって，$(i_{e_1},j_{e_1},k_{e_1},c_{e_1}),\ldots,(i_{e_m},j_{e_m},k_{e_m},c_{e_m})$ $((i_{e_t},j_{e_t},k_{e_t},c_{e_t}) \in E')$ をこの順に通るものが存在する必要十分条件は，

- $i_u \leq \lfloor i_{e_1} \rfloor, \lceil i_{e_m} \rceil \leq i_v, \lceil i_{e_t} \rceil \leq \lfloor i_{e_{t+1}} \rfloor$ $(1 \leq t \leq m-1)$
- $j_u \leq \lfloor j_{e_1} \rfloor, \lceil j_{e_m} \rceil \leq j_v, \lceil j_{e_t} \rceil \leq \lfloor j_{e_{t+1}} \rfloor$ $(1 \leq t \leq m-1)$
- $k_u \leq \lfloor k_{e_1} \rfloor, \lceil k_{e_m} \rceil \leq k_v, \lceil k_{e_t} \rceil \leq \lfloor k_{e_{t+1}} \rfloor$ $(1 \leq t \leq m-1)$

のすべてがみたされていることであるといえる．このときの経路のコストは $C(|i_u-i_v|+|j_u-j_v|+|k_u-k_v|)-\sum_t c_{e_t}$ であるから，上をみたすような辺の列について $\sum_t c_{e_t}$ の最大値を求めればよい

これらを踏まえて，

(a) 頂点 u を始点としたアルゴリズムにおいて $\mathrm{get}(j_v,k_v) = c_v$ のとき，$\sum_t c_{e_t} = c_v$ となるような条件をみたす列が存在する事

(b) $\sum_t c_{e_t} = c_v$ となるような列が存在するとき，頂点 u を始点としたアルゴリズムにおいて $\mathrm{get}(j_v,k_v) \geq c_v$ となる事

を示す．

3.3.1 (a) の証明

$\mathrm{get}(j_v,k_v) = c_v$ のとき，アルゴリズムにおいて更新操作が行われる履歴から，クエリの列 $(i_{q_1},j_{q_1},k_{q_1},-1,c_{q_1}),\ldots,(i_{q_m},j_{q_m},k_{q_m},-1,c_{q_m})$ であって，次をみたすものが

存在する．

- $(i_u, j_u, k_u, u, 0)$, 列 $(i_{q_1}, j_{q_1}, k_{q_1}, -1, c_{q_1})$, $\ldots$, $(i_{q_m}, j_{q_m}, k_{q_m}, -1, c_{q_m})$, $(i_v, j_v, k_v, v, 0)$ はこの順で (連続しているとは限らない) クエリ列 Q に含まれている．
- $j_u \leq |j_{q_1}|, \lceil |j_{q_m}| \rceil \leq j_v, \lceil |j_{q_t}| \rceil \leq \lfloor |j_{q_{t+1}}| \rfloor$ $(1 \leq t \leq m-1)$
- $k_u \leq |k_{q_1}|, \lceil |k_{q_m}| \rceil \leq |k_v|, \lceil |k_{q_t}| \rceil \leq \lfloor |k_{q_{t+1}}| \rfloor$ $(1 \leq t \leq m-1)$
- $\sum_{t=1}^{m} c_{q_t} = c_v$

j_u, k_u が整数である事から $j_u \leq \lfloor |j_{q_1}| \rfloor$, $k_u \leq \lfloor |k_{q_1}| \rfloor$ が成り立っていることに注意すると，$(i_{q_1}, j_{q_1}, k_{q_1}, -1, c_{q_1}), \ldots, (i_{q_m}, j_{q_m}, k_{q_m}, -1, c_{q_m})$ に対応する辺をこの順に通る最短経路が存在するには

- $i_u \leq \lfloor i_{q_1} \rfloor, \lceil i_{q_m} \rceil \leq i_v, \lceil i_{q_t} \rceil \leq \lfloor i_{q_{t+1}} \rfloor$ $(1 \leq t \leq m-1)$

が成り立っていれば良い．クエリの列が昇順でソートされていることから $i_u \leq i_{q_1} \leq \cdots \leq i_{q_m} \leq i_v$ が成り立つ．これと i_u, i_v が整数である事から $i_u \leq \lfloor i_{q_1} \rfloor$, $\lceil i_{q_m} \rceil \leq i_v$ は成り立つ．i_{q_t}, $i_{q_{t+1}}$ はともに整数または半整数であるから，$i_{q_t} \leq i_{q_{t+1}}$ かつ $\lceil i_{q_t} \rceil > \lfloor i_{q_{t+1}} \rfloor$ となるのは $i_{q_t} = i_{q_{t+1}}$ かつこれらが半整数の時のみである．しかし，このとき j_{q_t},$j_{q_{t+1}}$,k_{q_t},$k_{q_{t+1}}$ はいずれも整数であるから $|j_{q_t}| \leq |j_{q_{t+1}}|$ および $|k_{q_t}| \leq |k_{q_{t+1}}|$ が成り立ち，さらにこのとき j_{q_t},$j_{q_{t+1}}$,k_{q_t},$k_{q_{t+1}}$ はすべて負であるから $j_{q_t} \geq j_{q_{t+1}}$ および $k_{q_t} \geq k_{q_{t+1}}$ が成り立ち，クエリが昇順に並べられていることから $j_{q_t} = j_{q_{t+1}}$,$k_{q_t} = k_{q_{t+1}}$ となり，同一辺の情報が複数含まれていることになり，これはあり得ない．よって，$\lceil i_{q_t} \rceil \leq \lfloor i_{q_{t+1}} \rfloor$ となり，条件をみたす列が存在する事が分かる．

3.3.2 (b) の証明

辺の列 $(i_{e_1}, j_{e_1}, k_{e_1}, c_{e_1}), \ldots, (i_{e_m}, j_{e_m}, k_{e_m}, c_{e_m})$ であって，頂点 u から頂点 v までの最短経路であってその順で通る事が出来るようなものが与えられたとする．このとき，クエリの列において，頂点 u に対応するクエリの後に各辺に対応するクエリが辺の列の順に，そして最後に頂点 v に対応するクエリが現れる事を示す．$1 \leq t \leq m-1$ について，$\lceil i_{e_t} \rceil \leq \lfloor i_{e_{t+1}} \rfloor$ が成り立っているが，$i_{e_t} < i_{e_{t+1}}$ ならば e_t と e_{t+1} に対応するクエリはこの順で登場し．そうでないとき $i_{e_t} = i_{e_{t+1}}$ かつこれが整数であるから，両者に対応するクエリの j,k 要素は正で与えられる．このとき，$\lceil j_{e_t} \rceil \leq \lfloor j_{e_{t+1}} \rfloor$, $\lceil k_{e_t} \rceil \leq \lfloor k_{e_{t+1}} \rfloor$ よりやはり e_t と e_{t+1} に対応するクエリはこの順で現れる．頂点 u と辺 e_1, 辺 e_m と頂点 v についても同様の事が成り立つ．

また，$j_u \leq \lfloor j_{e_1} \rfloor$, $\lceil j_{e_t} \rceil < \lfloor j_{e_{t+1}} \rfloor$ $(1 \leq t \leq m-1)$, $k_u \leq \lfloor k_{e_1} \rfloor$, $\lceil k_{e_t} \rceil < \lfloor k_{e_{t+1}} \rfloor$ $(1 \leq t \leq m-1)$ より，$j_u \leq j_{e_1} \leq \cdots \leq j_{e_m}$, $k_u \leq k_{e_1} \leq \cdots \leq k_{e_m}$ であるから，更新操作はつねに行われる．このとき，帰納的に $\mathrm{get}(\lfloor j_{e_t} \rfloor, \lfloor k_{e_t} \rfloor) \geq \sum_{s=1}^{t-1} c_s$ が成り立ち，最終的に $\mathrm{get}(j_v, k_v) \geq \sum_{t=1}^{m} c_t$ が成り立つ．

よって，(a)(b) の両方が示され，アルゴリズムの正当性が示された．

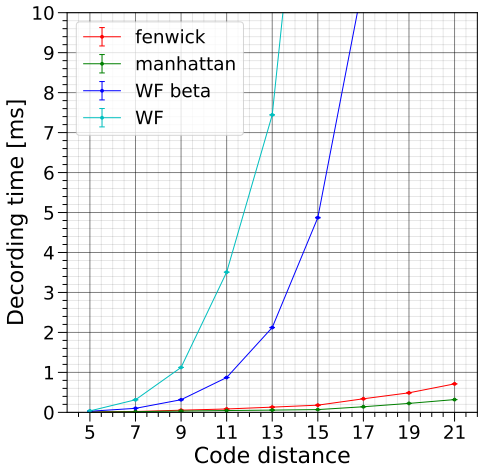
4. 数値計算による評価

4.1 数値計算の設定

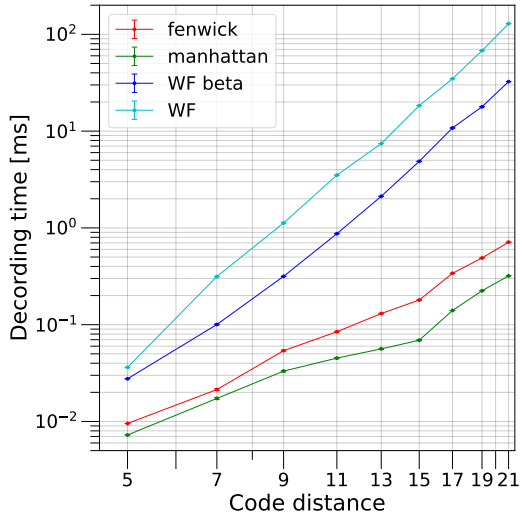
本手法をマンハッタン距離を直接計算する手法と，ワーシャルフロイド法により全点間距離を計算するアルゴリズムと比較を行った．実装は C++を用いて行った．コンパイルは gcc 9.2.0 を用いて行い，O2 オプションで最適化した．数値計算は Intel Core i5-8250U の CPU で行った．表面符号は X と Z のエラーについて対象であるため，パウリ X エラーを訂正する際の時間と誤り訂正の性能を評価した．また，今回は簡単のため測定エラーは無いものとして 2 次元の格子に対して数値計算を行った．数値計算では不均質なエラーモデルとして，以下のようなエラーモデルを採用している．まず，量子ビットそれぞれが確率 0.1 でエラーが発生しやすい量子ビットに変化する．その後，正常な量子ビットについては与えらえた確率 p で，エラー確率が発生しやすい量子ビットについては確率 0.5 でパウリ X エラーが発生するとしている．以降の数値計算のグラフでは各データ点に対して，10^5 回ずつサンプリングを行い，エラーバーとともにプロットを行っている．

4.2 速度

我々はまず，符号距離を $5 \leq d \leq 21$ の範囲で，正常な量子ビットのエラー率が $p = 0.01$ である場合において，まず復号に要した平均時間を計算した．その結果は**図 5** に線形および両対数でプロットされている．今回の提案手法は赤い線で表示されており，ベースラインとなるワーシャルフロイド法やその改善である WF beta に比べて小規模なサイズも含め実速度で高速であることが分かる．特に，実用化が有望視される $d = 21$ の場合においては，フェニック木を用いた提案手法はワーシャルフロイド法による手法の 180 倍の速度を達成している．本手法は全ての辺を均質な重みとみなすマンハッタン距離を用いた復号と比較すると低速であるが，その低速化の度合いはワーシャルフロイド法からの改善幅に比べれば軽微であると言える．特に，$d = 21$ の符号距離においては，提案手法は 2.2 倍の実行時間となっている．



(a) 速度比較



(b) 速度比較 (対数スケール)

図 5: 提案手法とベースラインとなる手法との速度比較

4.3 復号の性能

次に我々は復号の性能を比較した．本手法では複数の手法で 2 点間のノードの重みを計算し，計算された重みに基づいてエドモンドの花アルゴリズムを用いた復号を行った [16]．エドモンドの花アルゴリズムの実装には Kolmogorov らの実装を利用している [21]．$d = 7, 11, 15$ の場合について，復号の失敗確率，すなわち，論理エラーが生じる確率は**図 6** のように計算された．提案したフェニック木を用いた手法による復号の成功率はワーシャルフロイド法による手法とほぼ同等の性能を達成した．これはエラーを均一なものとみなしてマンハッタン距離を計算して復号する手法と比較して有意に優れている．

二次元平面の格子に対する復号アルゴリズムの性能は図 6 のように符号距離の拡大に際し相転移的なふるまいを示すことが知られており，十分大きな符号距離での性能は相転移点となるエラー率，すなわち，符号距離を変更したときの交点となるしきい値と呼ばれるエラー率で評価できる．この基準においては，しきい値が大きいほどエラーが大きな量子計算機でも誤り訂正できることになるので，しきい値が大きい復号アルゴリズムの方が優れていると判定することができる．

均質なエラーレートが実現されているときにエドモンドの花アルゴリズムで復号を行うと，表面符号のしきい値は約 10%となることが知られている．今回のエラーモデルでは低頻度でエラーの大きな量子ビットが存在するため，ワーシャルフロイド法で厳密に重みを計算したとしても，点線で表されるようにそのしきい値は 9%に低下している．一方，不均質さを無視して全てを均質なエラーとしてマンハッタン距離で重みを計算したケースでは，破線で表されるようにそのしきい値は 6%まで小さくなっており，論理エラー自体もワーシャルフロイド法に比べて悪化していることが分かる．今回の提案手法は実線で表示されており，論理エラーについてもしきい値についても，その性能はほぼワーシャルフロイド法の結果と等しいことが分かる．このことは，今回の研究で取り入れた近似は必ずしも最適な重みを与えるとは限らないものの，復号アルゴリズムの観点ではエラー推定の精度に殆ど悪影響を与えないことが分かる．従って，本研究の提案手法は殆ど論理エラーの性能を悪化させずに従来のアルゴリズムに比べ大幅な高速化を実現できていることが分かる．

5. まとめと展望

本研究では不均質なエラー率で誤りが生じる量子ビットの表面符号による誤り訂正において，復号アルゴリズムの重要なサブルーチンである 2 点間の最小重み経路の重みの計算を，誤り訂正の性能の劣化を殆どなしに高速化する手法を提案した．本研究では誤り訂正の復号の対象となる格子が 3 次元の正方格子をなすことに着目し，フェニック木を用いたデータ構造を導入することで従来に比べ計算量的な改善を実現している．また，具体的な実装を通して今回の高速化は興味のある領域において有効であることを示し，さらに符号のしきい値が殆ど劣化していないことを示した．従って，今回の研究結果は寿命の短さから高速な復号が求められる超伝導量子ビットなどを用いた誤り耐性量子計算の実現可能にするうえで重要なアルゴリズムであると言える．

本研究では簡単のために数値計算は測定エラーが無いと仮定した 2 次元格子についてのみ行ったが，測定エラーを考慮した 3 次元格子での性能評価を行うのがまず今後の研究として考えられる．また，実際の誤り耐性量子計算では格子手術 [6], [22] などを通して格子の形状が計算中に変化するため，こうした格子形状に変化に合わせてデータ構造を適応させることも今後の課題である．近年は正方格子ではないより疎なグラフにおいても高い性能を持つ符号が存

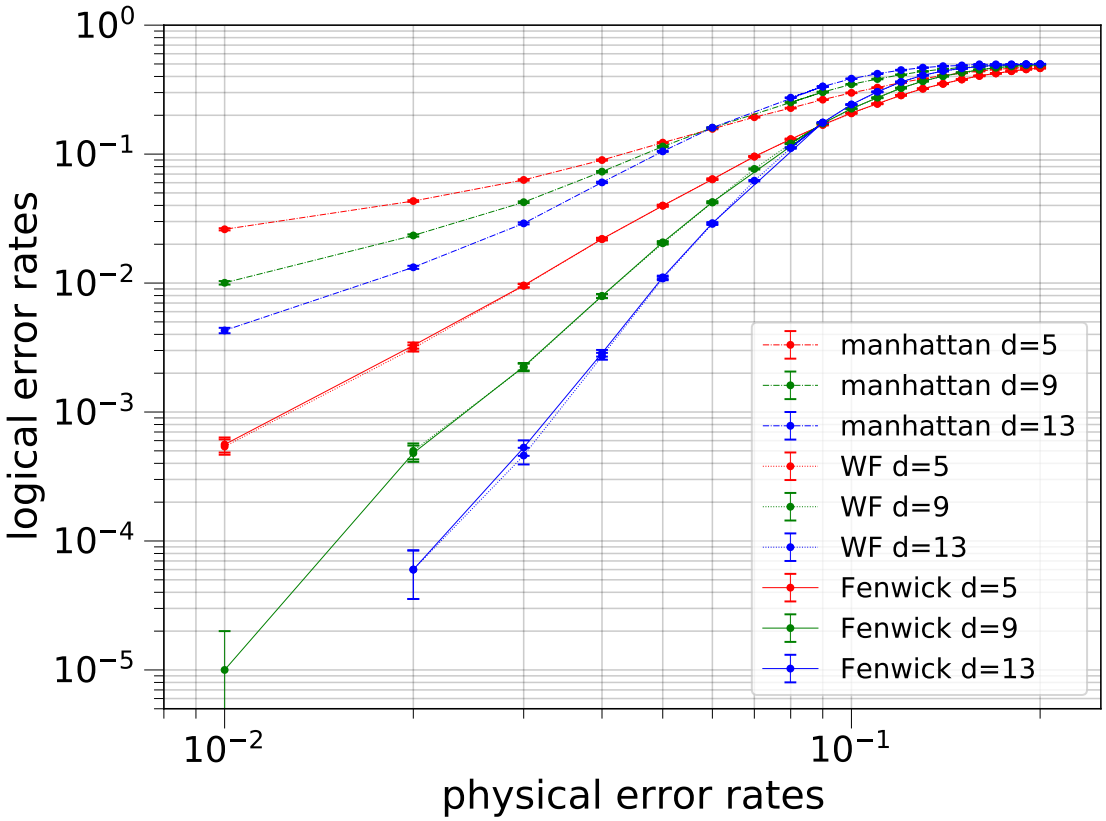


図 6: 復号性能の比較

在することが示されている [23]．こうした正方格子でないケースに対応できるよう本手法を拡張するのも興味深い．また，現実の量子誤り訂正ではスタビライザー測定で用いられる CNOT ゲートといった物理ゲートごとにエラーが生じると期待されるため，このような回路レベルでの不均質さを扱えるようにすることも実用的な性能をより正確に評価するうえで重要となる．

本研究では WF 法，WF beta 法，マンハッタン距離を計算する方法との比較を行ったが，他の手法との体系的な比較を行うことも今後の課題である．今回比較が出来なかったダイクストラ法に基づく方法のほか，従来知られているもので高速とされている不均質なエラーを扱える復号アルゴリズムとしては，Union-find データ構造を用いたものの拡張がある [9], [12]．本研究は Union-find データ構造のものと比べ計算量的な優位性があるかどうかはパラメータ領域に依存することが分かっている．Union-find データ構造を用いた復号アルゴリズムは C++での実装が公開されていないため今回の研究では比較を行わなかったが，具体的な実装と性能評価を通しこの二つの性能を比較する研究が今後の研究として期待される．

謝辞 本研究は JST さきがけ（助成番号：No. JPMJPR1916），内閣府ムーンショット（助成番号：No. JPMJMS2061）の助成の元で行いました．

参考文献

[1] Nielsen, M. A. and Chuang, I.: Quantum computation and quantum information (2002).
[2] Lidar, D. A. and Brun, T. A.: *Quantum error correction*, Cambridge University Press (2013).
[3] Kitaev, A. Y.: Quantum computations: algorithms and error correction, *Russian Mathematical Surveys*, Vol. 52, No. 6, pp. 1191–1249 (1997).
[4] Bravyi, S. B. and Kitaev, A. Y.: Quantum codes on a lattice with boundary, *arXiv preprint quant-ph/9811052* (1998).
[5] Fowler, A. G., Whiteside, A. C. and Hollenberg, L. C.: Towards practical classical processing for the surface code, *Physical review letters*, Vol. 108, No. 18, p. 180501 (2012).
[6] Fowler, A. G. and Gidney, C.: Low overhead quantum computation using lattice surgery, *arXiv preprint arXiv:1808.06709* (2018).
[7] Wang, C., Harrington, J. and Preskill, J.: Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory, *Annals of Physics*, Vol. 303, No. 1, pp. 31–58 (2003).
[8] Fowler, A. G., Sank, D., Kelly, J., Barends, R. and Martinis, J. M.: Scalable extraction of error models from the output of error detection circuits, *arXiv preprint arXiv:1405.1454* (2014).
[9] Delfosse, N. and Nickerson, N. H.: Almost-linear time decoding algorithm for topological codes, *arXiv preprint arXiv:1709.06218* (2017).
[10] Holmes, A., Jokar, M. R., Pasandi, G., Ding, Y., Pedram, M. and Chong, F. T.: NISQ+: Boosting quantum computing power by approximating quantum error

rection, *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, pp. 556–569 (2020).

[11] Ueno, Y., Kondo, M., Tanaka, M., Suzuki, Y. and Tabuchi, Y.: QECOOL: On-Line Quantum Error Correction with a Superconducting Decoder for Surface Code, *arXiv preprint arXiv:2103.07526* (2021).

[12] Das, P., Pattison, C. A., Manne, S., Carmean, D., Svore, K., Qureshi, M. and Delfosse, N.: A scalable decoder micro-architecture for fault-tolerant quantum computing, *arXiv preprint arXiv:2001.06598* (2020).

[13] Das, P., Locharla, A. and Jones, C.: LILLIPUT: A Lightweight Low-Latency Lookup-Table Based Decoder for Near-term Quantum Error Correction, *arXiv preprint arXiv:2108.06569* (2021).

[14] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C.: *Introduction to algorithms*, MIT press (2009).

[15] Fenwick, P. M.: A new data structure for cumulative frequency tables, *Software: Practice and experience*, Vol. 24, No. 3, pp. 327–336 (1994).

[16] Edmonds, J.: Paths, trees, and flowers, *Canadian Journal of mathematics*, Vol. 17, No. 3, pp. 449–467 (1965).

[17] Chamberland, C., Noh, K., Arrangoiz-Arriola, P., Campbell, E. T., Hann, C. T., Iverson, J., Putterman, H., Bohdanowicz, T. C., Flammia, S. T., Keller, A., Refael, G., Preskill, J., Jiang, L., Safavi-Naeini, A. H., Painter, O. and Brandão, F. G.: Building a Fault-Tolerant Quantum Computer Using Concatenated Cat Codes, *PRX Quantum*, Vol. 3, No. 1 (online), DOI: 10.1103/prxquantum.3.010329 (2022).

[18] Nagayama, S., Fowler, A. G., Horsman, D., Devitt, S. J. and Van Meter, R.: Surface code error correction on a defective lattice, *New Journal of Physics*, Vol. 19, No. 2, p. 023050 (2017).

[19] McEwen, M., Faoro, L., Arya, K., Dunsworth, A., Huang, T., Kim, S., Burkett, B., Fowler, A., Arute, F., Bardin, J. C., Bengtsson, A., Bilmes, A., Buckley, B. B., Bushnell, N., Chen, Z., Collins, R., Demura, S., Derk, A. R., Erickson, C., Giustina, M., Harrington, S. D., Hong, S., Jeffrey, E., Kelly, J., Klimov, P. V., Kostritsa, F., Laptev, P., Locharla, A., Mi, X., Miao, K. C., Montazeri, S., Mutus, J., Naaman, O., Neeley, M., Neill, C., Opremcak, A., Quintana, C., Redd, N., Roushan, P., Sank, D., Satzinger, K. J., Shvarts, V., White, T., Yao, Z. J., Yeh, P., Yoo, J., Chen, Y., Smelyanskiy, V., Martinis, J. M., Neven, H., Megrant, A., Ioffe, L. and Barends, R.: Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits, *arXiv preprint arXiv:2104.05219* (2021).

[20] Pattison, C. A., Beverland, M. E., da Silva, M. P. and Delfosse, N.: Improved quantum error correction using soft information, *arXiv preprint arXiv:2107.13589* (2021).

[21] Kolmogorov, V.: Blossom V: a new implementation of a minimum cost perfect matching algorithm, *Mathematical Programming Computation*, Vol. 1, No. 1, pp. 43–67 (2009).

[22] Horsman, C., Fowler, A. G., Devitt, S. and Van Meter, R.: Surface code quantum computing by lattice surgery, *New Journal of Physics*, Vol. 14, No. 12, p. 123011 (2012).

[23] Gidney, C., Newman, M., Fowler, A. and Broughton, M.: A fault-tolerant honeycomb memory, *Quantum*, Vol. 5, p. 605 (2021).