可逆メモリ管理のための小石並べゲーム

2019SE045 小椋響　2019SE054 清水岳
指導教員：横山　哲郎

1　はじめに

量子コンピュータには量子論理ゲートを組み合わせた量子回路を用いて行うものがある [7]．本研究は，こうしたゲート型を対象とする．量子回路では中間結果の削除という非可逆的な操作を避けるため，中間結果の可逆的な処理が必要であり，処理の方法は量子ビットの数や量子演算の回数によって，多様な組み合わせが存在する．その結果，量子ビットの数や量子演算の回数といったコストは組み合わせ毎に異なることとなる．量子ビットの実現にはコストが大きいため，中間結果の可逆的な処理に関する効率的なメモリ管理が必要とされている．

二項演算から構成される式を計算する量子回路は二分木の構造をもつ．このことを利用して，本研究では量子回路を二分木における小石並べゲームに対応付ける．限られた量子ビット数で中間結果の可逆的な処理が要求されているという量子回路の問題は，限られた石の数で中間結果の出力を行わないという共通点を持つ小石並べゲームとして考えることができる．小石並べゲームは Bennett によって紹介された．量子ビット数を小石の数に，計算時間をステップ数に対応させることで小石並べゲームの最適解がそのまま可逆メモリ管理における中間結果の最適化問題につながる．そのため二分木における小石並べゲームの使う石の数とステップ数を指標としたアルゴリズムの最適化を目的とし，本研究では完全二分木という特殊なケースについてを考える．完全二分木における使う石の数とステップ数を指標としたアルゴリズムの最適化を行う．

2　関連研究

2.1　可逆計算

コンピュータは目的の出力を行う過程で情報の削除を行っている．この情報の削除によって熱が発生し，エネルギー消費に繋がっている [3]．可逆計算はこの熱エネルギーを発生させない計算方法であり，単射の性質も持っている．また，可逆計算は可逆性の性質を有する量子回路にも使われている．可逆計算には中間結果の可逆的な処理が必要である．これは中間結果を保存した状態で目的の出力を行い，計算の終了時点から逆計算を行うことで中間結果を元の状態に戻している [1, 6]．これによって情報の削除で発生するエネルギー消費をなくすことができる．

2.2　量子回路

従来のコンピュータでは，ビットが存在し，「0」と「1」で表される．このような 2 進数の演算ではどちらか一方をとるため同時に 2 以上とることはない．量子ビットでは，量子力学の重ね合わせという性質を用いることでどちらも取ることができる．一般的に $\alpha|0\rangle + |1\rangle$ と表される重ね合わせの量子状態のことを量子ビットという [5]．二項演算から構成される式を計算する量子回路は二分木で表される [4]．二項演算から構成される式の最適化についてはできないものも存在する．本研究では，二分木で表される量子回路に関する最適化を目的としている．二分木と量子回路の対応は図 1である．

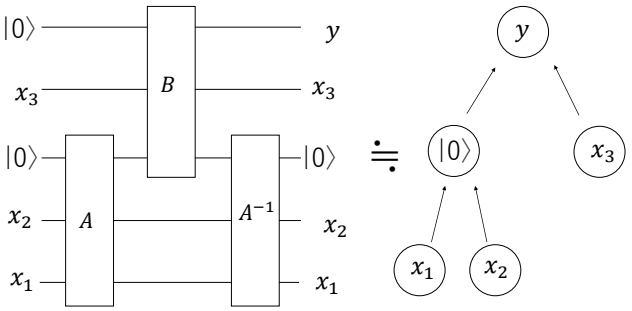

図 1　量子回路と二分木の対応

3　小石並べゲーム

小石並べゲームではクリアまでの高さを T，使うことのできる石の数を S，クリアまでのステップ数を T'，各手順の段数を k，手順に再帰を使用した回数を n とする．マスが 1 から T 個まで縦に並べられた図 2のようなボードを使用する．

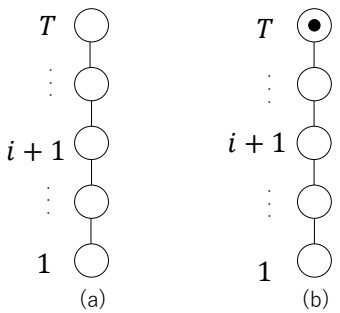

図 2　丸いマスが 1 から T 個縦に並べられたボード

マス i $(0 < i < T - 1)$ には石が置かれているもしくは空であるかのいずれかの状態である．マス i に石が置かれている状態なら，1 段上の丸いマス $i+1$ に新たな石を追加

するか置かれている石を削除することができる．ただし，丸いマス 1 には無条件に石を追加，削除することもできる．図 2の (a) の初期状態では, 1 から T のすべてのマスに石が置かれてない状態である．最終的にマス T のみに石がある状態である (b) の状態になればクリアである．クリアまでの石を追加したり削除した手順のまとめたものを棋譜と呼び，マス T 以外のマスに存在する石を中間結果と呼ぶ.

Bennett は小石並べゲームを紹介すると同時に，2 つの方法を提案した．1 つ目は，1 から T マスまで一つずつ石を積み上げていく 1973 年の方法であり，B1973 とする．この方法では，ステップ数を少なくすることができるので最短でクリア可能だが使う石の数が多くなる．2 つ目は，定数 k と n を与えることで手順を再帰的に繰り返し，使う石の数，最高点，ゲームのステップ数を導ける 1989 年の方法である [2]．この方法を B1989 とする．B1989 では，n が同じ構成で k が増加すると，より少ない時間でクリアできるものの，多く石を使用してしまう．k を変数とした場合の B1989 におけるクリアまでの高さ T, クリアまでのステップ数 T', 使うことの出来る石の数 S' は次のようになる.

$$T(k,0) = 1 \tag{1}$$
$$T(k,n+1) = kT(k,n) \qquad (n \geq 0) \tag{2}$$
$$T'(k,0) = 1 \tag{3}$$
$$T'(k,n+1) = (2k-1)T'(k,n) \qquad (n \geq 0) \tag{4}$$
$$S'(k,0) = 1 \tag{5}$$
$$S'(k,n+1) = S'(k,n) + (k-1) \qquad (n \geq 0) \tag{6}$$

4　完全二分木における小石並べゲーム

本稿では，小石並べゲームを完全二分木上で実現しており，ルールが次のように変更される．まず，対象となる完全二分木の葉には自由に石の追加と削除を行うことができる．葉以外のノードに対しては，子ノードの 2 つに石が置かれている場合に親ノードの石を追加で置くか削除することができる．これらの動作を繰り返し行い，最終的に根にのみ石が存在する状態になるとゲームクリアとなる．d を深さとし，我々は 5 つの方法の提案を行う.

4.1　B1973 の方法の拡張

1 つ目の方法は B1973 の方法を拡張したものである．この方法を B1973T とする．深さ d における使える石の数を $S'_1(d)$，ステップ数を $T'_1(d)$，ノード数を $M(d)$ とするとアルゴリズムは algorithm1のようになる.

$$t ::= L \mid C \;\; c \;\; t \;\; t$$
$$c \in (0,1)$$

0 は石が存在しない，1 は石が存在する.

Algorithm 1 B1973T

```
1:  procedure B1973T_SUB(t)
2:      if t = L then
3:          return
4:      else
5:          let C c t1 t2 = t
6:          B1973T_SUB(t1)
7:          B1973T_SUB(t2)
8:          c ← 1 − c
9:      end if
10: end procedure
11:
12: procedure B1973T(t)
13:     B1973T_SUB(t)
14:     先頭の石をコピーして保持する
15:     B1973T_SUB^{-1}(t)
16: end procedure
```

$$M(d) = 2^{d+1} - 1 \tag{7}$$
$$T'_1(d) = 2M(d) - 1 \tag{8}$$
$$S'_1(d) = M(d) \tag{9}$$

4.2　B1989 の方法の拡張

B1989 の方法の拡張を B1989T とする．B1989T では完全二分木を使い，小石並べゲームの 1 マスを同じ深さのノードと考えて B1989 を適用することで拡張としている．完全二分木の深さを d とすると $d = k^n - 1$ になり，完全二分木のノードの数 M は $M = 2^{d+1} - 1$ となると考えられる．また，この方法によってたどり着ける深さを $d(k,n)$, 変数を $x(k,n)$，使える石の数を S'_2，ステップ数を T'_2 とし，$n \geq 0$ とする．0 は石が存在しない，1 は石が存在する．アルゴリズムは algorithm2のようになる.

$$d(k,0) = 0 \tag{10}$$
$$d(k,n+1) = kd(k,n) + 1 \tag{11}$$
$$x(k,0) = 2 \tag{12}$$
$$x(k,n+1) = x^k(k,n) \tag{13}$$
$$T'_2(k,0) = 1 \tag{14}$$
$$T'_2(k,n+1) = (1 + 2\sum_{a=2}^{k} x^{a-1}(k,n))T'_2(k,n) \tag{15}$$
$$S'_2(k,n) = \sum_{a=0}^{k-2} 2^{d(k,n)-a} + 4\sum_{b=1}^{n} 2^{d(k,n)+1-k^b} \tag{16}$$

4.2.1　再帰を用いたアルゴリズム

この再帰を用いるアルゴリズムを B1989R とする．最初に深さを 1 ずつ増やす B1989 の方法について考える．$BT'(d)$ をステップ数，$BS'(d)$ は石の数，Bd はこの方法での深さとする．B1989 年の方法における $(k,\ n) = (2, n-1)$ を用いて深さ $d(2, n-1)$ 進める．進むべき残り

Algorithm 2 完全二分木上の 1989 年の方法

procedure B1989T_SUB(t)
  $d \leftarrow t$ の深さ
  $depth(t) = d + 1$
  if $t = L$ **then**
    return
  else
    $depth(t)/2$ より下の $2^{(depth(t)+1)/2}$ 個の二分木 t_i $(0 \leq i \leq 2^{(depth(t)+1)/2} - 1)$ について
    B1989T_SUB(t_i)
    let $let\ C\ c\ t1'\ t2' = t_i$
    $c \leftarrow 1 - c$
    B1989T_SUB^{-1}(t_1)
    $depth(t)/2$ より先の部分リストを L に置き換えたリスト t' について
    B1989T_SUB(t')
  end if
end procedure

procedure B1989L(t)
  B1989T_SUB(t)
  先頭の石をコピーして保持する
  B1989T_SUB^{-1}(t)
end procedure

の深さが $Bd - d(2, n-1) - 1$ に対して深さを 1 ずつ増やす B1989 の方法を深さ $Bd - d(2, n-1) - 1$ で適用し逆順で中間結果の削除を行う．ステップ数 BT' と必要な石の数 BS' は次のようになると考えられる．

$$d(2,\ n-1) < Bd < d(2,\ n) \tag{17}$$
$$BT'(0) = 1 \tag{18}$$
$$BT'(Bd) = 2T'(2,\ n-1) + BT'(Bd - d(2,\ n-1) - 1) \tag{19}$$
$$BS'(Bd) = S'(2,\ n) \tag{20}$$

この深さを 1 ずつ増やす B1989 の方法と再帰を用いて解いていく．$T'_3(d)$ をステップ数，$S'_3(d)$ を石の数として図 3のように深さ $d-1$ の完全二分木の方法を再帰する．同様に深さ $d-2$ から深さ 0 まで順番に再帰を行っていく．再帰を適用していくと下の図 3に示してあるような状態になる．そして図 3の枠に囲まれていないノードに注目すると深さを 1 ずつ増やす B1989 の方法で深さ d の完全二分木の根まで到達可能である．

B1989 の方法の適用で使う石の数を $BS'(d)$ とすると深さ d の完全二分木のクリアに必要な石の数は $d + BS'(d)$ となると考えられる．最後に手順の逆順を行う．このような手順で行われる再帰を用いたアルゴリズムのステップ数

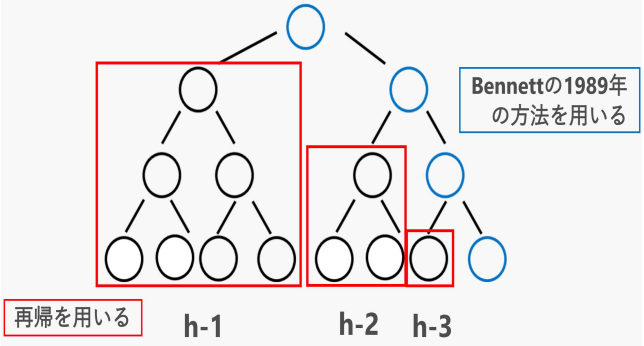

図 3　再帰と Bennett の方法の適用

T'_3 と必要な石の数 S'_3 は次のようになると考えられる．

$$T'_3(0) = 1 \tag{21}$$
$$T'_3(d+1) = 3T'_3(d) + BT'(d+1) - BT'(d) \tag{22}$$
$$S'_3(d) = d + BS'(d) \tag{23}$$

4.2.2　分割統治法

分割統治法は，ある問題に対して同じ構造を持つより小さな問題に分割する方法であり，多くのアルゴリズムで使われている．

分割統治法と B1973T を用いたアルゴリズム　Bennett の 1973 年の方法の拡張と分割統治法を用いる方法である．この方法を B1973T-d とする．T'_5 をステップ数，S'_5 を石の数とする．根の持つ子ノードを部分木とみると完全二分木となる．この分割した左部分木と右部分木の完全二分木を B1973T を用いることでクリアする方法である．

ステップ数 T'_5 と必要な石の数 S'_5 は次のようになると考えられる．

$$T'_5(0) = 1 \tag{24}$$
$$T'_5(1) = 5 \tag{25}$$
$$T'_5(d+1) = T'_5(d) + 2^{d+3} \quad (d \geq 1) \quad S'_5(0) = 1 \tag{26}$$
$$S'_5(1) = 3 \tag{27}$$
$$S'_5(d+1) = 2S'_5(d) - 1 \quad (d \geq 1) \tag{28}$$
$$\tag{29}$$

この漸化式を解くと，$T'_5 = 2^{d+3} - 11 (d \geq 1)$，$S'_5 = 2^d + 1 (d \geq 1)$ を得ることができる．

分割統治法を用いたアルゴリズム　深さ 0，深さ 1 と順番に解いていくことによって元の問題を解くことができる．T'_6 をステップ数，S'_6 を石の数，d を深さとし，このアルゴリズムを d-Tree とする．図 4の完全二分木では，枠をそれぞれ深さ 0，深さ 1，深さ 2 の完全二分木として見ることができる．深さ 0 を再帰的に用いることで深さ 1 を解くことができ，分割した問題を再帰的に用いることで元の問題を解くことができる．

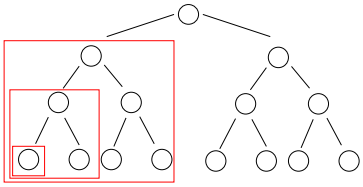

図 4　d-Tree

この方法のステップ数 T'_6 と必要な石の数 S'_6 は次のようになると考えられる．

$$T'_6(0) = 1 \tag{30}$$
$$T'_6(d+1) = 4T'_6(d) + 1 \quad (d \geq 0) \tag{31}$$
$$S'_6(0) = 1 \tag{32}$$
$$S'_6(d+1) = S'_6(d) + 2 \quad (d \geq 0) \tag{33}$$

上記の二つのようにステップ数と石の数の漸化式が得られる．この漸化式を解くと，$T' = \frac{4(4^n-1)}{3} + 1$，$S' = 2n + 1$ を得ることができる．

5　おわりに

それぞれのアルゴリズムのステップ数と石の数を比較したものを図 5，図 6に示す．Bennett の 1973 年の拡張法ではステップ数が最小に，使う石の数は最大になる方法であると考えられる．一方で，Bennett の 1989 年の拡張法は少ない石の数で実現可能だが，ノード数が指数関数で増えていくために多くの石を必要とするため，少ない石の数でクリア可能な新たなアルゴリズムが必要である考えられる．再帰を用いたアルゴリズムは石の数の最適化を目指して提案を行った．その結果，図 6からも分かるように他の方法と比較しても少ない石の数でクリア可能であると考えられる．本研究では，有限の大きさの完全二分木を対象としているので，各アルゴリズムによってクリアできること，及びクリアまで有限ステップで停止できることは直観にあう．しかし，各アルゴリズムに関するクリアまでの深さ d，使うことのできる石の数 S，クリアまでのステップ数 T' などの値が正しいかは検証が必要である．我々は，深さ d が 15 以下の完全二分木についての棋譜を手計算して一般項の値と一致することを確かめた．帰納法によって一般にこれらが一致するか否かを示すことは今後の課題である．

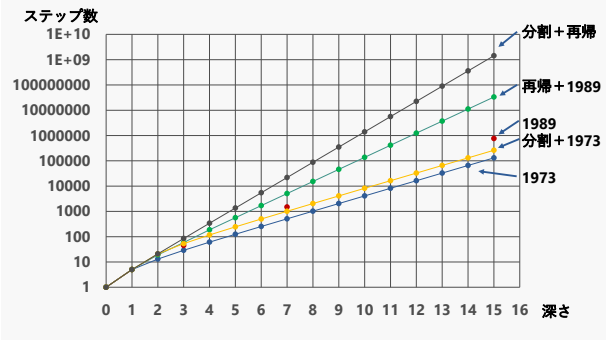

図 5　ステップ数

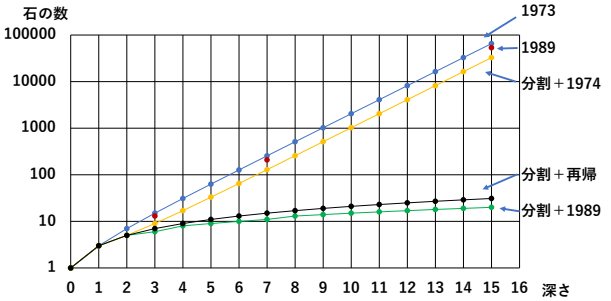

図 6　石の数

参考文献

[1] Bennett, C. H.: Logical Reversibility of Computation, *IBM Journal of Research and Development*, Vol. 17, No. 6, pp. 525–532 (1973).

[2] Bennett, C. H.: Time/Space Trade-Offs for Reversible Computation, *SIAM Journal on Computing*, Vol. 18, pp. 766–776 (1989).

[3] Landauer, R.: Irreversibility and heat generation in the computing proces, *IBM Journal of Research and Development*, Vol. 5, pp. 183–191 (1961).

[4] Meuli, G., Soeken, M., Roetteler, M., Bjorner, N. and Micheli, G. D.: Reversible Pebbling Game for Quantum Memory Management, *2019 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 288–291 (2019).

[5] 細谷暁夫：量子コンピュータの基礎，サイエンス社 (1999).

[6] 森田憲一：可逆計算，近代科学社 (2012).

[7] 西野哲朗：量子コンピュータ入門，東京電機大学出版局 (1997).