社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

量子遺伝的操作の改良

齋藤　暁†　　ラヒミ　ロバベ††　　中原　幹夫†††

† 近畿大学総合理工学研究科量子コンピュータ研究センター
〒 577-8502 大阪府東大阪市小若江３－４－１
†† Institute for Quantum Computing, University of Waterloo,
200 University Ave. West, Waterloo, Ontario, N2L 3G1, Canada
††† 近畿大学理工学部理学科物理学コース
〒 577-8502 大阪府東大阪市小若江３－４－１
† 現所属：国立情報学研究所情報学プリンシプル研究系量子情報理論研究室
〒 101-8430 東京都千代田区一ツ橋２－１－２
E-mail: †akirasaitoh@nii.ac.jp

あらまし　我々は文字通りの意味での量子遺伝的アルゴリズムでは初めて、量子交叉を含むアルゴリズムを提案した [arXiv: 1202.2026 (cs.NE)]。各遺伝的操作で量子論的並列性を活用するために、既存の量子遺伝的アルゴリズムとは一線を画す技術的な改良を施した。例えば、Grover 操作で使う平均に関する折り返し操作のコストの軽減のために擬似スクランブラーを導入するなどした。本発表ではそれら改良点を解説する。
キーワード　量子遺伝的アルゴリズム, 計算量

Improvements in quantum genetic operations

Akira SAITOH†, Robabeh RAHIMI††, and Mikio NAKAHARA†††

† Research Center for Quantum Computing, Interdisciplinary Graduate School of Science and Engineering,
Kinki University, 3-4-1 Kowakae, Higashi-Osaka, Osaka 577-8502, Japan
†† Institute for Quantum Computing, University of Waterloo,
200 University Ave. West, Waterloo, Ontario, N2L 3G1, Canada
††† Department of Physics, Kinki University,
3-4-1 Kowakae, Higashi-Osaka, Osaka 577-8502, Japan
† Present address: Quantum Information Science Theory Group, National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430, Japan
E-mail: †akirasaitoh@nii.ac.jp

Abstract　We have recently developed a quantum genetic algorithm which has both quantum crossover and mutation operations [arXiv: 1202.2026 (cs.NE)]. Improvements have been made on each of genetic operations so that they are substantially different from those of existing quantum genetic algorithms. For instance, we have tailored the inversion-about-average operation of the Grover routine by introducing a pseudo-scrambler so as to reduce the internal cost. In this talk, we are going to present the details of improvements.
Key words　Quantum genetic algorithm, Computational complexity

1. はじめに

量子風の古典遺伝的アルゴリズム [1]〜[9] が活発に開発されてきたのに比べて、文字通りの量子遺伝的アルゴリズムはあまり注目を集めず、これまでに少数のアルゴリズム [10]〜[15] が提案されただけであった。また、それらのいずれも、全ての遺伝的操作に量子論的並列性が利用されているわけではなかった。特に、量子交叉は議論されてはいたが [11], [13]、構築方法の提案はされていなかった。

遺伝的アルゴリズムは、数値的な最適化アルゴリズムの一つである。典型的な解くべき問題は、以下のようなものである。

$0, \ldots, N-1$ のインデックス i が割り当てられた N 個体の染色体 x_i がある。染色体の適応度 $\in [0,1]$ を計算する関数 f が与えられている（ここでは適応度を規格化している）。適応度 $f(x_i)$ が十分大きくなる x_i を見つけよ。

通常、N は各個体の染色体の長さ n に比べて指数的に大きいので、総当たりで探索するのは困難である。なお、適応度関数は、内部計算量は小さいが最適な入力を見つけるのが困難な関数を想定している [16]。例えば、節の数が C 個ある Conjunctive Normal Form (CNF) が与えられたとして、ビット列で表される染色体を入力としたとき、B 個の節を充足するとする。すると、充足した割合 B/C をその染色体 (を持つ個体) の適応度とすることができる。

遺伝的アルゴリズムの操作は、(i) 選択、(ii) 交叉、(iii) 突然変異に大別される [16], [17]。初期世代として、適当な染色体の集合を考える。例えば、ランダムに生成した $\widetilde{N} < N$ 染色体の集合を使う。通常、$\widetilde{N} \ll N$ である。ここでは、各個体が、n ビット長の 2 進文字列の染色体一つで表されている場合を考えることにする。この初期世代から始め、各世代で (i)-(iii) の操作を行って次の世代に適応度が高い染色体の個体数が多くなるように繰り返すことで、最終的に十分適応度が高い染色体を持つ個体のインデックスを出力することができる。

(i) 選択操作は、ある世代に含まれる染色体のうち、適応度が大きいものを優先的に次の世代に繰り越すための操作であり、適応度が大きな染色体の個体数を増大させ、適応度が小さな染色体の個体数を減少させる。

この選択操作を量子的に行うために、既存の量子遺伝的アルゴリズム [11]〜[15] はすべて、Grover 探索 [18] の派生アルゴリズムである Grover-BBHT 探索 [19] あるいはそれを用いた最大を探索するアルゴリズム [20], [21] を使用している。しかし、いずれのアルゴリズムでも、Grover 探索で使われる折り返し操作の内部構成が言及されていない。

Grover の繰り返し操作には、オラクル関数に基づく操作 (オラクル操作) と母集団の平均状態に関する折り返し操作 (折り返し操作) が使われる。適応度関数の内部計算量は小さいという前提 [16] があるので、$f(x_i)$ がある閾値より大きな染色体をターゲット状態としてそれらの係数の符号を反転するオラクル操作には、$\mathrm{poly}(\log N)$ の回路量しか要さない。しかし、折り返し操作は、各世代の選択操作開始時点での集団に含まれるインデックスがばらけているため、単に Hadamard 変換を $1 - 2|0\rangle\langle 0|$ に作用させるだけでは構成できない。

ある t 代めの世代での選択開始前の集団 G_t に対応する重ね合わせ状態を $|s_t\rangle = \sum_{x \in G_t} c_x |x\rangle$ (ここで染色体 x の占有数を $|c_x|^2$ とする。ただし、$c_x \in \mathbb{C}$ は $\sum_x |c_x|^2 = 1$ を満たす。) とする。この世代の選択操作のために Grover 探索で使われる折り返し操作は、$1 - 2|s_t\rangle\langle s_t|$ であるが、これは通例 [22] では適当な量子ゲート U_t を使って $U_t(1 - 2|0\rangle\langle 0|)U_t^\dagger$ の形で構成する。状態 $|0\rangle$ を状態 $|s_t\rangle$ に写す U_t の構成は、現在知られている最も回路量の小さな構成方法 [23] を用いても、一般には回路量 $O(\sharp G_t \log N)$ を要する。(なお、Soklakov と Schack [24] はこれが $\mathrm{poly}(\log N)$ 資源でできるという誤解を生む主張をしているが、彼らのアルゴリズムの内部回路量を正しく評価すれば、実は $O(\sharp G_t \mathrm{poly}(\log N))$ 回路量を使っていることが分かる。) ゆえに、既存の量子アルゴリズムの選択操作は、Grover 探索を用いているにも関わらず、たとい $O(1)$ クエリーしか消費しない特別な場合 [12], [14] でも、$O(\sharp G_t \log N)$ 回路量を要する。古典的な遺伝的アルゴリズムの選択にかかるコストと同じオーダーのコストを使っていることになる。なお、$\sharp G_t = N$(文献 [11], [13] での設定) であれば U_t として $H^{\otimes n}$ が使えるので選択操作に $O(\sqrt{N}\mathrm{poly}(\log N))$ 資源しかかからないが、遺伝的アルゴリズムを使うのは $\sharp G_t$ が小さくても良いためであるから、折り返し操作のコストを小さくするために $\sharp G_t = N$ とするのは本末転倒である。

(ii) 交叉は、集団から複数の個体を取り出して、染色体の部分文字列を交換してから戻す操作である。もともとの $\widetilde{N}$ 個の染色体の集合の中で探索したのでは、探索すべき空間のごく一部しか見ないことになる。交叉を行うことで、探索空間が広くなる。本稿では染色体 2 つずつをペアにして交叉を行う場合を考える。

量子遺伝的アルゴリズムでは、世代 t の開始時点での集団を前出の重ね合わせ状態 $|s_t\rangle$ として持っているので、一見、交叉を並列化できそうである。しかし、文字列を二つ指定してその部分文字列を交換するという操作は、結局多数の制御ビットを使う制御操作になってしまい、一度に複数の交叉を行うことができない。量子並列性を交叉の並列化に使う議論はあったが、具体的にアルゴリズムが提案されてはいなかった [11], [13]。

(iii) 突然変異は、集団から染色体を適宜取り出して、その適当な場所の文字を変化させて戻す操作である。本稿の設定では適当な桁のビットを反転させる操作になる。突然変異の目的は、交叉と同じく、探索できる空間の拡大である。

量子遺伝的アルゴリズムにおいて、少数の制御付きビットフリップを使って、制御ビットによって指定される多数の染色体に同じ突然変異を同時に行うことは容易である。この点を明示的に利用したアルゴリズム [15] がある (ただし、やはり量子交叉は含んでいない)。

以上のように、既存の量子遺伝的アルゴリズムでは、選択操作で Grover 探索の能力が発揮できず、さらに、量子交叉の具体的提案がなされていなかった。我々は、各遺伝的操作で量子論的並列性が使われるように改良したアルゴリズム [25] を提案する。まず、各世代の開始状態の生成に擬似的スクランブラーを写像として使うという工夫により、折り返し操作の内部コストを多項式コストに抑え、選択操作において Grover 探索の能力が発揮されるようにする。次に、同一レジスタを 2 つ用意し、量子ビットのラベルの割り振りの変更だけで簡便に量子交叉ができることを示す。また、テンプレートに基づく量子突然変異を提示する。最後に、提案するアルゴリズムとそれに対応する古典アルゴリズムを計算量について比較評価する。

2. 量子遺伝的操作の改良

各遺伝的操作で量子論的並列性を使う我々のアルゴリズムを以下に示す。まずアルゴリズムの全容を述べ、続いて各遺伝的操作の詳細を説明する。

2.1 アルゴリズムの全容

※各染色体は長さ n のビット列である。また、正の整数 $c < n$ があり、$\widetilde{N} = 2^c \lessapprox N = 2^n$ であると想定している。

Algorithm 1

十分に大きな適応度とみなす閾値 $f_{\rm th}$ を定める。
$t \longleftarrow 0$.
REPEAT 1.-9.:

1. c ビット長ビット列を n ビット長擬似ランダムビット列に写す写像として、擬似的スクランブラー R を構築する。これは、入出力がそれぞれ $(a, 0)$ と $(a, R(a))$ である、可逆な $\mathrm{poly}(cn)$ ゲートで構成される回路として構築する (ここで a は c ビット長ビット列である)。R は入力 $0_0 \cdots 0_{c-1}$ を例外として $0_0 \cdots 0_{n-1}$ に写すものとする。この R は、次の t まで変更せずに使う。具体的な構築方法は 2.2 節で説明する。
2. **IF** $t = 0$ **THEN** ランダムな c ビット長ビット列 γ を生成し、 set $z \longleftarrow R(\gamma)$. (なお $t \neq 0$ であれば、z は $(t-1)$ 世代の最終段階で選抜された最適な染色体が保持されている。) **ENDIF**
3. ランダムな c ビット長ビット列 γ' を生成し、set $u \longleftarrow R(\gamma')$.
4. **CALL** `init_reg`$(R,\ z)$ (定義は 2.2 節) を 2 回実行し、同一の状態 2 つを得る。その状態それぞれが $|\varphi\rangle = \frac{1}{\sqrt{N}} \sum_{x \in X} |a_x\rangle_{\rm a} |x\rangle$ であり、a_x は $x \in X$ へのポインターアドレス、ただし $X = \{R(q)\}_q \cup \{z\}$ (ここで $q = 0_0 \cdots 0_{c-2} 1_{c-1}, \ldots, 1_0 \cdots 1_{c-1}$) である。なお、下付き文字 "a" はアドレス部分を表している。このプロセス全体をユニタリー変換 $U_{\rm init}$ と書く。すなわち、$|\varphi\rangle^{\otimes 2} = U_{\rm init}(|0\rangle_{\rm a}|0\rangle)^{\otimes 2}$ である。
5. **CALL** `quantum_crossover`(l) (定義は 2.3 節). これは、現在の量子レジスタの重ね合わせ状態を成している状態の全ての組み合わせに対する 1 点交叉であり、交叉点は $(l-1)$ 番目のビットと l 番目のビットの間にある。ただし、$|\varphi\rangle^{\otimes 2}$ に作用する操作としては、恒等写像である。
6. 量子突然変異 (定義は 2.4 節) を量子レジスタに適用する。このプロセス全体をユニタリー変換 $U_{\rm mut}$ と書く。
まったく同じ突然変異を古典的操作として u に適用する。
7. **CALL** `quantum_selection`$(U_{\rm init},\ U_{\rm mut},\ u)$ (定義は 2.5 節) を量子レジスタに適用し、この関数からの出力である染色体 z を得る。
8. **IF** $f(z) \geqq f_{\rm th}$ **THEN RETURN** z and **EXIT ENDIF**
9. $t \longleftarrow t + 1$.

量子レジスタをリフレッシュする。
END REPEAT

2.2 各世代の初期状態生成

ここではプロシージャ `init_reg`$(R,\ z)$ を定義する。これは、その世代の開始時点での個体集団を生成するプロシージャであり、擬似スクランブラー R を使って z 以外の個体染色体をランダムに生成する。出力は z および生成されたランダム染色体の重ね合わせ状態 $|\varphi\rangle = \frac{1}{\sqrt{N}} \sum_{x \in X} |a_x\rangle_{\rm a} |x\rangle$ (ここで $X = \{R(q)\}_q \cup \{z\}$) である。

PROCEDURE `init_reg`$(R,\ z)$:

(i) $|0\rangle_{\rm a}^{\otimes c} |0\rangle^{\otimes n}$ に $H^{\otimes c} \otimes I^{\otimes n}$ を適用し、状態 $\frac{1}{\sqrt{\sharp X}} \sum_{j=0}^{2^c-1} |j\rangle_{\rm a} |0 \cdots 0\rangle$ を得る。

(ii) 擬似スクランブラー R を適用する。R : $|j\rangle_{\rm a}|0\rangle \mapsto |j\rangle_{\rm a}|x_j\rangle$ はユニタリー変換であり、x_j は n ビット擬似乱数である $(j = 1, \ldots, 2^c - 1)$。ただし仮定により、$|0\rangle_{\rm a}|0\rangle$ は $|0\rangle_{\rm a}|0\rangle$ に写される。具体的な R の構成は下で述べる。

(iii) $0_0 \cdots 0_{c-1}$-controlled $X^{z_0} \otimes \cdots \otimes X^{z_{n-1}}$ を適用して、$|0\rangle_{\rm a}|0\rangle$ を $|0\rangle_{\rm a}|z\rangle$ に写す。ここで、z_k は z の k ビット目を表す $(k = 0, \ldots, n-1)$。

(iv) **RETURN** 現在の状態、つまり $|\varphi\rangle$.

R の具体的な構成方法:

擬似スクランブラー R は以下のように構成できる。$(c+n)$ 量子ビットを考える。$i \in \{0, \ldots, c-1\}$ に対して、ビット i を制御ビットとして、ランダムテンプレート T_i にしたがって標的ビット $c, \ldots, c+n-1$ をビット反転する。ここで T_i は 100101100100 のような、ランダムに生成された n ビット長ビット列であり、1 がビット反転、0 が無反転に対応する。これを i が 0 から $c-1$ まで繰り返す。この構成が擬似スクランブラーとしての使用に耐えることは、文献 [25] の補遺で示した。

このようにして構成した R の回路量は、$O(cn)$ である。R の出力乱数の性質を改善する目的でより複雑な構成にしても、回路量は $\mathrm{poly}(cn)$ で十分であるはずである。

2.3 交　　叉

ここではプロシージャ `quantum_crossover`(l) を定義する。これは、交差点がビット $l-1$ と l の間にある 1 点交叉を、考え得るすべてのペアについて並列化したものである。プロシージャが呼ばれる直前の量子レジスタは以下のような、同じ状態 2 つの直積状態にある。

$$|\varphi\rangle^{\otimes 2} = \left(\frac{1}{\sqrt{\sharp X}} \sum_{x \in X} |a_x\rangle_{\rm a} |x^{\rm left}\rangle |x^{\rm right}\rangle \right) \otimes \left(\frac{1}{\sqrt{\sharp X}} \sum_{x' \in X} |a_{x'}\rangle_{\rm a} |x'^{\rm left}\rangle |x'^{\rm right}\rangle \right).$$

ここで、x の左側 l ビットを $x_{\rm left}$、右側 $n-l$ ビットを $x_{\rm right}$

と書いた。この状態で単純に中間の部分 $|x^{\rm right}\rangle|x'^{\rm left}\rangle$ を考えなければ、すべての組み合わせの交叉が完了している。中間部分を $|*_{xx'}\rangle$ と書く。また、$|x^{\rm left}x'^{\rm right}\rangle$ を主要部分 (main) とみなす。すると、

$$\frac{1}{\sharp X}\sum_{x\in X}\sum_{x'\in X}|a_xa_{x'}\rangle_{\rm a}|x^{\rm left}x'^{\rm right}\rangle_{\rm main}|*_{xx'}\rangle. \qquad (1)$$

を得る。これで、すべての可能な組み合わせについての 1 点交叉後の染色体と、元の染色体が含まれた重ね合わせ状態になっている。つまり、交叉のためには量子ビットラベルを読み替えるだけで良い。

PROCEDURE `quantum_crossover`(l):

(i) 状態 $|\varphi\rangle^{\otimes 2}$ がある。元の量子ビットのラベルが $0,\ldots,2c+2n-1$ であるとする。ラベルを次のように振り直す：$\underbrace{0,\ldots,c-1}_{c}$, $\underbrace{2c,\ldots,2c+l-1}_{l}$, $\underbrace{2c+n,\ldots,2c+2n-l-1}_{n-l}$, $\underbrace{c,\ldots,2c-1}_{c}$, $\underbrace{2c+2n-l,\ldots,2c+2n-1}_{l}$, $\underbrace{2c+l,\ldots,2c+n-1}_{n-l}$.

(ii) **RETURN**

2.4 突 然 変 異

ここでは、プロシージャ`tmp_mut` を定義する。これは、テンプレートに基づく突然変異である。

PROCEDURE `tmp_mut`:

(i) 突然変異を起こすべきスキーマを指定するテンプレート、例えば`***0*1***1**0****1*`を作る。これはランダムに作っても良い。これによって、集団の中の染色体のうち、指定された場所に 0,1,1,0,1 があるものを指定することになる。

(ii) 二つ目のテンプレート、例えば`*X**********X***` を作る。ここで、 `X` は一つ目のテンプレートでは`*`が置かれていた場所にのみ置ける。(i) で指定した染色体それぞれの、このテンプレートで `X` が置かれたビットについてビット反転を行なう。

(iii) **RETURN**

Note: 今までの世代で見つかった最適な染色体である z を変化させたくない場合は、z が除外されるように一つ目のテンプレートを選べば良い。

量子回路としては、この例では、"**0-controlled 1-controlled 1-controlled 0-controlled 1-controlled NOT NOT**" を、テンプレートで指定した場所に従って配置すれば良い。要するに、二つのテンプレートを適宜作って、それらに従って複数制御ビット付き複数 NOT ゲートを実行するだけである。

突然変異操作前の状態は式 (1) である。突然変異操作実行後の染色体を $(\widetilde{x}^{\rm left}\widetilde{x'}^{\rm right})$ と書けば、全体の状態は突然変異操作によって、

$$|\widetilde{\varphi}\rangle=\frac{1}{\sharp X}\sum_{x\in X}\sum_{x'\in X}|a_xa_{x'}\rangle_{\rm a}|\widetilde{x}^{\rm left}\widetilde{x'}^{\rm right}\rangle_{\rm main}|*_{xx'}\rangle \qquad (2)$$

になる。

2.5 選　　択

ここでは、プロシージャ`quantum_selection`($U_{\rm init}$, $U_{\rm mut}$, u) を定義する。これは、染色体集団 $\{(\widetilde{x}^{\rm left}\widetilde{x'}^{\rm right})\}$ の中で最も適応度 $f(\widetilde{x}^{\rm left}\widetilde{x'}^{\rm right})$ が大きな染色体を選択する操作である。

この直前の状態は式 (2) である。なお、整数定数 $\eta\geqq 1$ を使って、繰り返し回数 $k_{\rm term}=\eta\times\lceil(45/2)\widetilde{N}+(28/5)(\log_2\widetilde{N})^2\rceil$ を与えておく。

PROCEDURE `quantum_selection`($U_{\rm init}$, $U_{\rm mut}$, u):

FOR $k\longleftarrow 0$ **TO** $k_{\rm term}-1$:

(i) ユニタリー変換

$$U_1=I\otimes I\otimes I-2\sum_{f(y)\geqq f(u)}I\otimes|y\rangle\langle y|\otimes I$$

を構築する。ここで左右の I はそれぞれアドレス部分の状態 $|a_xa_{x'}\rangle$ および状態 $|*_{xx'}\rangle$ に作用する。U_1 はオラクル操作であり、次のように構築できる。

まず、量子レジスタに補助量子ビットブロック (I) と (II) を取り付け、 その状態を $|0\cdots0\rangle_{\rm (I)}|-\rangle_{\rm (II)}$ にする。ここで $|-\rangle=(|0\rangle-|1\rangle)/\sqrt{2}$ である。個々の $\widetilde{x}^{\rm left}\widetilde{x'}^{\rm right}$ の適応度 $f(\widetilde{x}^{\rm left}\widetilde{x'}^{\rm right})$ をブロック (I) の値として設定する。 これは f を量子回路として作用させればできる。 この操作を U_f と書くことにする。そしてブロック (I) を、値 $f(u)$ と比較する。もし $f(\widetilde{x}^{\rm left}\widetilde{x'}^{\rm right})\geqq f(u)$ ならば量子ビット (II) をビットフリップする。操作 $U_f{}^\dagger$ を行い (これで補助量子ビットブロックと量子レジスタのエンタングルメントが解ける)、補助量子ビットブロックを取り外す。

(ii) ユニタリー変換

$$U_2=I\otimes I\otimes I-2|\widetilde{\varphi}\rangle\langle\widetilde{\varphi}|$$

を構築する。これは折り返し操作であり、以下のように構築できる。

$$U_2=U_{\rm mut}U_{\rm init}[I-2(|0\rangle^{\otimes c}|0\rangle^{\otimes n}\langle 0|^{\otimes c}\langle 0|^{\otimes n})^{\otimes 2}]U_{\rm init}^\dagger U_{\rm mut}^\dagger.$$

ただし、`quantum_crossover`(l) に従って量子ビットラベルの割り振りも変えている。

(iii) Grover-BBHT 探索 [19] を呼び出す。ただし、標的状態の符号反転をするオラクル操作と、平均に関する折り返し操作を、それぞれ、U_1 と U_2 で置き換えて使用する。また、探索開始時の状態は $|\widetilde{\varphi}\rangle$ を使う。

(iv) レジスタの main 部分を測定し、染色体 u' を得る。

IF $f(u')>f(u)$ **THEN** $u\longleftarrow u'$ **ENDIF**

END FOR

RETURN u.

ここで、繰り返し回数 $k_{\rm term}$ の説明をする。上のプロシージャが行なうことは、関数出力を最大にする入力を求めるための量子探

索 [20], [21] を、部分空間 $\mathrm{span}\{|a_x a_{x'}\rangle_{\mathrm{a}}|\widetilde{x}^{\mathrm{left}}\widetilde{x'}^{\mathrm{right}}\rangle_{\mathrm{main}}|*_{xx'}\rangle\}$ で実行することに他ならない。Dürr と Høyer [20] が証明したように、もし繰り返し回数が $\lceil(45/2)\sqrt{M}+(7/5)(\log_2 M)^2\rceil$ であれば、関数出力を最大化する入力が見つかる確率は少なくとも 1/2 である。ここで、M はインデックスの数である。我々は、現在のプロシージャの実行前に交叉を実行していたため、$\widetilde{N}^2$ の異なるアドレス $a_x a_{x'}$ が存在する。従って、k_{term} 回の繰り返しの後では、染色体集団 $\{(\widetilde{x}^{\mathrm{left}}\widetilde{x'}^{\mathrm{right}})\}$ の中では適応度が最大となる染色体が見つかる確率が、少なくとも $1-(1/2)^{\eta}$ になる。ゆえに、η を適当な数、例えば 16 程度にすると、ほぼ確実にその染色体集団の中で適応度最大の染色体が得られる。

3. 計算量の比較評価

我々のアルゴリズムの計算量を示し、対応する古典アルゴリズムの計算量と比較する。

3.1 計算量評価

各遺伝的操作ごとの計算量は以下の通りである。

各世代の初期個体集団生成のコスト:

2.2 節から、各世代の初期個体集団生成には、$\mathrm{poly}(cn) = \mathrm{poly}(\log\widetilde{N}\log N)$ 個の基本量子ゲートを使用することが分かる。

交叉のコスト:

2.3 節から、全ての染色体の組み合わせについて 1 点交叉を考えるのに必要な操作は、古典的なラベルの付け替えだけである。つまり $O(\log N)$ 時間しか要さない。

突然変異のコスト:

2.4 節で述べたように、突然変異は複数制御ビットで制御して複数標的ビットをフリップする操作であるので、$O(\log N)$ 回路量しかかからない。

選択操作のコスト:

2.5 節で詳述した選択操作では、交叉の結果発生する $\widetilde{N}^2$ 個の染色体から最も適応度の高い染色体を選ぶために、$O(\widetilde{N})$ クエリーを要する。そのクエリー 1 回ごとに U_1 と U_2 が呼び出される。適応度関数の内部計算量は $\mathrm{poly}(n)$ が想定されている [16] ので、U_1 は $\mathrm{poly}(n)$ 回路量で構成できることになる。U_2 は内部で U_{init} と U_{mut}、およびそれらの逆変換を呼び出すが、たかだか $\mathrm{poly}(\log\widetilde{N}\log N)$ 回路量しか要しない。ゆえに、選択操作の時間量は $O(\widetilde{N}\mathrm{poly}(\log\widetilde{N}\log N))$ である。空間量は、適応度関数の想定される内部空間量 $\mathrm{poly}(\log N)$ のオーダーを越えない。

以上を総合して、一世代あたりの計算量は、時間量が、基本ゲートの数とオーダーが一致するとして、$O(\widetilde{N}\mathrm{poly}(\log\widetilde{N}\log N))$、また、空間量が $\mathrm{poly}(\log N)$ となる。

3.2 対応する古典アルゴリズム

上述の我々の量子遺伝的アルゴリズム (Algorithm 1) と同じ働きをする古典アルゴリズムは以下のようになる。Algorithm 1 と同様に、各染色体は長さ n のビット列であり、また、正の整数 $c<n$ があり、$\widetilde{N}=2^c \lneqq N=2^n$ であると想定している。

Algorithm 2

適応度が十分大きいとみなす閾値 f_{th} を定める。

$t \longleftarrow 0$.

REPEAT 1.-8.:

1. 擬似スクランブラー $R:\{0,1\}^c \to \{0,1\}^n$ を古典的な論理演算を使って構成する。2.2 節で述べた構成方法がそのまま使える。
2. 2.4 節で導入した突然変異を写像 M として、古典的な論理演算で構成する。.
3. **IF** $t=0$ **THEN** ランダムに $\gamma\in\{1,\ldots,\widetilde{N}-1\}$ を一つ取ってきて、$z \longleftarrow R(\gamma)$. (なお $t\neq 0$ の場合は、z には $(t-1)$ 世代の最終段階で選抜された最適な染色体が保持されている。) **ENDIF**
4. ランダムに $\gamma'\in\{1,\ldots,\widetilde{N}-1\}$ を一つ取ってきて、$j \longleftarrow R(\gamma')$.
5.

FOR $a \longleftarrow 0$ **TO** $\widetilde{N}-1$:
  IF $a=0$ **THEN** $x \longleftarrow z$ **ELSE** $x \longleftarrow R(a)$ **ENDIF**
  FOR $b \longleftarrow 0$ **TO** $\widetilde{N}-1$:
    IF $b=0$ **THEN** $y \longleftarrow z$ **ELSE** $y \longleftarrow R(b)$ **ENDIF**
    x と y を交叉させ、交叉後の染色体 v と w を得る。
    染色体 $M(x)$, $M(y)$, $M(v)$, $M(w)$ の中で最も適応度が高いもの g を得る。
    IF $f(g)>f(j)$ **THEN** $j \longleftarrow g$ **ENDIF**
  END FOR
END FOR

6. $z \longleftarrow j$.
7. **IF** $f(z)\geqq f_{\mathrm{th}}$ **THEN RETURN** z and **EXIT ENDIF**
8. $t \longleftarrow t+1$.

END REPEAT

このアルゴリズムでは各 t で、内部コストが $\mathrm{poly}(\log\widetilde{N}\log N)$ である R が $O(\widetilde{N}^2)$ 回呼ばれている。よってこのアルゴリズムは各世代について、$O(\widetilde{N}^2\mathrm{poly}(\log\widetilde{N}\log N))$ 時間を消費する。空間量については、$\mathrm{poly}(\log N)$ 幅を使用する。

3.3 比　　較

我々が提案した量子遺伝的アルゴリズム (Algorithm 1) の一世代あたりの時間量は $O(\widetilde{N}\mathrm{poly}(\log\widetilde{N}\log N))$ であり、対応する古典アルゴリズム (Algorithm 2) の一世代あたりの時間量 $O(\widetilde{N}^2\mathrm{poly}(\log\widetilde{N}\log N))$ と比べて因子 $\widetilde{N}$ だけ小さい。空間量は両方ともに $\mathrm{poly}(\log N)$ である。

Algorithm 2 の時間量については、古典演算を使って交叉ですべての組み合わせを試しているので、$\widetilde{N}^2$ という因子が現れるのは必然と言える。この因子が Algorithm 1 では $\widetilde{N}$ になるのは Grover-BBHT 探索を用いる以上当然のように一見して思えるが、決して当然の帰結ではない。量子探索の能力が発揮

されているのは、平均に関する折り返し操作の内部コストが $\log \widetilde{N} \log N$ の多項式で抑えられているためであり、すなわち、擬似スクランブラーを導入した成果である。

4. むすび

各遺伝的操作を改良した量子遺伝的アルゴリズムを構築した。まず選択操作については、そもそも既存の量子遺伝的アルゴリズムでは内部コストが正しく考慮されていなかった。我々は Grover 操作における平均に関する折り返し操作を多項式回路量で構成する方法を導入し、量子選択操作として妥当な量子性に基づく高速化を実現した。これには擬似スクランブラーの写像としての導入が貢献している。また、これまでは量子論的並列化の議論はあったものの古典交叉が使われていた交叉操作について、量子交叉を明示的に導入した(注1)。さらに、テンプレートに基づく量子突然変異の構成方法も示した。

量子 “風” ではない、文字通りの量子遺伝的アルゴリズムの研究は活発ではない。今後、近年様々に拡張されてきた古典遺伝的アルゴリズムそれぞれに対応する量子アルゴリズムが研究されることが望まれる。

文　献

[1] A. Narayanan and M. Moore. Quantum-inspired genetic algorithms. In *Proceedings of the IEEE 3rd International Conference on Evolutionary Computation (ICEC96)*, pages 61–66, Nagoya, Japan, 20-22 May 1996. IEEE Press, Piscataway, NJ, 1996.

[2] K.-H. Han and J.-H. Kim. Genetic quantum algorithm and its application to combinatorial optimization problem. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC2000)*, pages 1354–1360, La Jolla, CA, 16-19 July 2000. IEEE Press, Piscataway, NJ, 2000.

[3] K.-H. Han and J.-H. Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Trans. Evol. Comput.*, 6(6):580–593, 2002.

[4] K.-H. Han and J.-H. Kim. Quantum-inspired evolutionary algorithms with a new termination criterion, h_ϵ gate, and two-phase scheme. *IEEE Trans. Evol. Comput.*, 8(2):156–169, 2004.

[5] S. Nakayama, T. Imabeppu, S. Ono, and I. Iimura. Consideration on pair swap strategy in quantum-inspired evolutionary algorithm. *IEICE Trans. Inf. Sys.*, J89-D(9):2134–2139, 2006. in Japanese.

[6] S. Nakayama, T. Imabeppu, and S. Ono. Pair swap strategy in quantum-inspired evolutionary algorithm, 2006. in the Late-breaking papers of the 2006 Genetic and Evolutionary Computation Conference (GECCO-2006), Seattle, WA, 8-12 July 2006.

[7] M. Chen and H. Quan. Quantum-inspired evolutionary algorithm based on estimation of distribution. In *Proceedings of the 2nd International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2007)*, pages 17–19, Zhengzhou, China, 14-17 September 2007. IEEE Press, Piscataway, NJ, 2007.

[8] R. Liao, X. Wang, and Z. Qin. A novel quantum-inspired genetic algorithm with expanded solution space. In *Proceedings of the 2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC 2010)*, pages 192–195, Nanjing, China, 26-28 August 2010. IEEE Computer Society, Los Alamitos, CA, 2010.

[9] G. Zhang. Quantum-inspired evolutionary algorithms: a survey and empirical study, 2010. J. Heuristics, online first, 29 June 2010, 49 pages, DOI:10.1007/s10732-010-9136-0.

[10] B. Rylander, T. Soule, J. Foster, and J. Alves-Foss. Quantum evolutionary programming. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1005–1011, San Francisco, CA, 7-11 July 2001. Morgan Kaufmann, San Francisco, 2001.

[11] M. Udrescu, L. Prodan, and M. Vlăduţiu. Grover's algorithm and the evolutionary approach of quantum computation, 2004. ACSA Report, "Politehnica" University of Timisoara, 15 Oct. 2004, http://www.acsa.upt.ro/publications/index.htm.

[12] A. Malossini, E. Blanzieri, and T. Calarco. QGA: quantum genetic algorithm, 2004. Technical Report: #DIT-04-105, Dec. 2004, Univ. Trento, http://www.dit.unitn.it.

[13] M. Udrescu, L. Prodan, and M. Vlăduţiu. Implementing quantum genetic algorithms: A solution based on Grover's algorithm. In *Proceedings of the 3rd Conference on Computing Frontiers*, pages 71–81, Ischia, Italy, 3-5 May 2006. ACM Press, New York, 2006.

[14] A. Malossini, E. Blanzieri, and T. Calarco. Quantum genetic optimization. *IEEE Trans. Evol. Comput.*, 12(2):231–241, 2008.

[15] D. Johannsen, P. P. Kuru, and J. Lengler. Can quantum search accelerate evolutionary algorithms? In M. Pelikan and J. Branke, editors, *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference (GECCO-2010)*, pages 1433–1440, Portland, OR, 7-11 July 2010. ACM, New York, NY, 2010.

[16] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[17] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. The University of Michigan Press, Ann Arbor, MI, 1975.

[18] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC 1996)*, pages 212–219, Philadelphia, PA, 22-24 May 1996. ACM Press, New York, 1996.

[19] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschr. Phys.*, 46:493–505, 1998.

[20] C. Dürr and P. Høyer. A quantum algorithm for finding the minimum, 1996. arXiv:quant-ph/9607014.

[21] A. Ahuja and S. Kapoor. A quantum algorithm for finding the maximum, 1999. arXiv:quant-ph/9911082.

[22] G. Brassard, P. Høyer, and A. Tapp. Quantum counting. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *Proceedings of Automata, Languages and Programming, 25th International Colloquium (ICALP'98)* (LNCS 1443), pages 820–831, Aalborg, Denmark, 13-17 July 1998. Springer-Verlag, Berlin, 1998. arXiv:quant-ph/9805082.

[23] D. Ventura and T. Martinez. Initializing the amplitude distribution of a quantum state. *Found. Phys. Lett.*, 12:547–559, 1999.

[24] A. N. Soklakov and R. Schack. Efficient state preparation for a register of quantum bits. *Phys. Rev. A*, 73:012307–1–13, 2006.

[25] A. SaiToh, R. Rahimi, and M. Nakahara. Semiclassical genetic algorithm with quantum crossover and mutation operations, 2012. arXiv:1202.2026.

(注1)：なお、量子交叉の操作は、本稿では１点交叉を考えたが、テンプレートに基づく交叉に置き換えるのは容易である。