並列化した遺伝アルゴリズムの拡張GCD算法への応用

2Q-01　松尾 直史 (愛媛大学大学院理工学研究科情報工学専攻)
甲斐 博 (愛媛大学工学部情報工学科)、 野田 松太郎 (愛媛大学工学部情報工学科)

1. はじめに

遺伝アルゴリズムは、選択淘汰、突然変異等の生物進化の原理に着想を得た、確率的探索、最適化の手法である。本研究では、遺伝アルゴリズムの新しい応用分野の開発を目指し、このアルゴリズムの多数の整数の最大公約数 (GCD) を求める問題への適用することを考える。Piehl 等は、拡張 GCD 算法で得られる係数を最小に近づける問題において、遺伝アルゴリズムによる方法を提案してる [1]。しかし、与える初期値によっては得られる結果が最小に近づかないことがある。また、多くの GCD 計算を必要とするため、計算時間は遅い。

本論では、以上の 2 つの問題に対し、遺伝アルゴリズムによる Sorting-GCD method の並列化を行うことを考え、より高速に最小に近い結果を与える並列アルゴリズムの提案を行う。

2. 拡張 GCD 算法

入力整数 $a = (a_1, a_2, \cdots, a_n)$ とする。本論で扱う拡張 GCD 算法は、n 個の整数の GCD $g = \mathrm{GCD}(a_1, a_2, \cdots, a_n)$ に対し、

$$g = \sum_{i=1}^{n} a_i x_i$$

となる整数 $x = (x_1, x_2, \cdots, x_n)$ を求める問題のことを指す。但し、解 y は一般に数多く存在する。

[1] では、ノルム $||x||$ を確率的に最小に近づける方法が遺伝アルゴリズムを基礎としていくつか提案されており、実験により比較が行われている。中でも Sorting-GCD method を用いた遺伝的アルゴリズムが最も $||x||$ を小さくしている。通常の Sorting-GCD method[2] は以下のような算法である。

Extended GCD Problem using Parallel Genetic Algorithm,
Naofumi Matsuo, Hiroshi Kai and Matu-Tarow Noda,
Ehime University,
3 Bunkyo-cho, Matsuyama, Ehime 790-8577, Japan.

アルゴリズム (Sorting-GCD method)
入力: $a = (a_1, a_2, \cdots, a_n)$
出力: $\sum_{i=1}^{n} a_i x_i = \mathrm{GCD}(a)$ を満足する
$x = (x_1, x_2, \cdots, x_n)$
< 方法 >
手順 0. n 次元ベクトル $d = (d_1, \cdots, d_n)$ を定義し、$d_k = a_{\pi(k)}, k = 1, \cdots, n$ と置く。ここで π は 1 から n までのランダムな順列とする。また行列 B を $B = (b_{ij}) = I_n$ のように単位行列とする。
手順 1. ベクトル d の要素の中で、d_i を最大値とし、d_j を 2 番目に大きな値とする。
手順 2. もし $d_j = 0$ ならば、$x_k = b_{i\pi^{-1}(k)}, k = 1, \cdots, n$ とし、このアルゴリズムを終了する。
手順 3. $d_i = d_i - d_j, b_{ik} = b_{ik} - b_{jk}$ を $\lfloor d_i/d_j \rfloor$ 回繰り返し、**手順 1.** に戻る。

このアルゴリズムによる解 x は、π の順列の与え方により異なる。そこで、π を染色体として与え、遺伝的アルゴリズムにより最適な π を得る方法が提案された。その方法を次節で示す。

3 遺伝的アルゴリズムと問題点

Sorting-GCD method を用いた遺伝的アルゴリズムの概要は次のように表される。以下、この方法を Genetic Sorting-GCD method と表す。

アルゴリズム (Genetic Sorting-GCD method)
入力: $a = (a_1, a_2, \cdots, a_n)$
出力: $\sum_{i=1}^{n} a_i x_i = \mathrm{GCD}(a)$ を満足する
$x = (x_1, x_2, \cdots, x_n)$
< 方法 >
手順 1. 染色体 (順列 π) を設計する。
手順 2. 初期集団 (順列 π の集合) を発生させる。
手順 3. 初期集団を適応値 (ノルム $||x||$) により評価する。(Sorting-GCD method を用いて x を求める)
手順 4. 適応値が高ものは生き残り、それ意外は淘汰される。

手順 5. 生き残った染色体をお互いに交叉する。染色体に突然変異を起こさせる。

手順 6. あらかじめ決められた時間が経過したら停止する。そうでなければ、**手順 3.** に戻る。

この方法によれば通常の Sorting-GCD method よりもノルムがより最小に近い解が得られる。しかし、与える染色体によっては、最小値に近づくのに時間がかかる場合がある。図 1 の例は、$n = 1000$ とし、e^{20} までの大きさの正の整数を a_i の値としてランダムに与えた場合である。

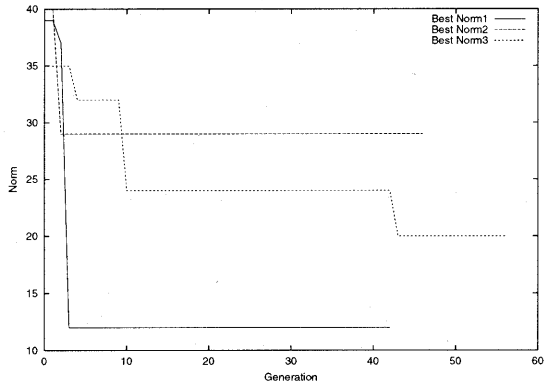

図 1: 異なる初期集団により得られるノルム値

縦軸は x の L1 ノルムを表し、横軸は遺伝回数を表す。同じ入力に対し 3 つの独立した遺伝を行った結果、初期集団の違いにより結果が大きく異なることが図からわかる。

4 遺伝的アルゴリズムの並列化

我々はこの問題に対し、並列計算機の各プロセッサ毎に遺伝的アルゴリズムを独立して進化させいく方法を取った。この時、各遺伝的アルゴリズムは、それぞれが独自の進化をとげているため、各世代における個体は各遺伝アルゴリズム毎に異なるものとなる。これらの独自に進化している個体群を、ある世代において、いずれかの遺伝アルゴリズムとの間で個体の交換を行う。これは Genetic Sorting-GCD method の**手順 6.** から**手順 3.** に戻る時に、プロセス間通信にて行われる。進化過程の異なる新たな個体を得ることにより、遺伝における多様性をもたせている。

以下の例は、$n = 1000$、a_i として $1 \sim e^{20}$ の値をランダムに与えた時の結果である。遺伝回数を 100 回として得られた L1 ノルムの値を示す。この結果より、プロセッサ数を増やすことにより、

表 1: プロセッサ数とノルムの値の関係

プロセッサ数	ノルム
1	17.1
2	15.8
4	14.8
6	14.4

ノルムの値を小さくできることが確認できる。

次にこれを高速化することに着目する。遺伝的アルゴリズムの実行には、入力ベクトルの要素数を増やせば時間がかかる。そこで、独立して進行している 1 つの遺伝的アルゴリズムにおいて**手順 3.** に複数のプロセッサを割り当て、Sorting-GCD method を並列して実行し、アルゴリズムを高速に実行できる。結果は表 2 に示す。

表 2: 高速化による影響

GA の数	高速化	ノルム	時間 (s)
6	1	14.4	8.901699
3	2	15.2	5.002896
1	6	17.1	2.759890

GA の数は独立した遺伝的アルゴリズムを同時に実行するプロセッサ数を表し、高速化は Sorting-GCD method を並列に実行するプロセッサ数を表す。以上の計算は並列計算機 Fujitsu AP3000 上で行った。

5 まとめ

Sorting-GCD method の問題点は、並列化することによる遺伝効果の向上、計算時間の短縮という手法を用いることによって補うことが可能であるという結果が得られた。

このような手法の今後の応用については、係数が整数に限ることなく浮動小数点でも同一の結果が予想され、数値数式融合処理の新しい発展に寄与するものと思われる。

参考文献

[1] Valerie Piehl, Jonathan P.Sorenson, Neil Tiedeman : Genetic Algorithms for the Extended GCD Problem, Butler University,1998.

[2] Bohdan S. Majewski and George Havas. A solution to the extended GCD problem, Proc.ISSAC'95,pp.248-253,1995.