遺伝的アルゴリズムの探索効率化に関する考察

8D−3

日高東潮　高井昌彰　佐藤義治
(北海道大学 工学部)

1 はじめに

遺伝的アルゴリズム (Genetic Algorithm 以下 GA と略す) は selection(淘汰), crossover(交叉), mutation(突然変異) の基本3操作を繰り返し適用することで、適応度関数と呼ばれる関数の最大化問題を近似的に解く最適化アルゴリズムである [1]。

これまでに数多くの改良された GA モデルが提案されてきたが、その多くが生物の進化モデルに依存しており、最適化アルゴリズムとしては冗長な部分が多く見られた。その中でも遺伝子集団内に同一遺伝子がかなりの重複度を持って存在する点が、有限である遺伝子集団の利用効率を考える上で非常に不利だと思われる。

そこで本研究では Simple GA(以下 SGA) をベースに遺伝子の重複を防ぐことで解空間探索の方法に改良を加えた GA のモデルを提案し、その有効性を実験により確認した。

今回提案するモデルは、個々の遺伝子にある種の属性を与えることにより、GA における探索を遺伝子間重複の出ない状態で行なおうというものである。

2 背景

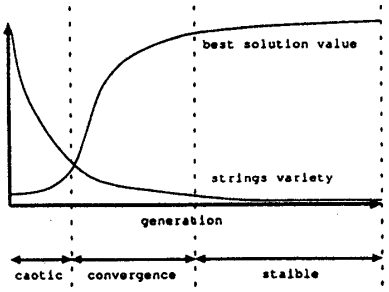

Figure.1

Figuer.1 は通常の GA における遺伝子の多様性の推移を示したものである。

これまでの SGA に関する実験等から、遺伝子が比較的乱雑な初期の状態から一様になるにつれ、適応度関数値に有効に働く schema が固定され、そのあと残りの部分が急速に収束することが分かっている [1] [2]。

このことより、初期の乱雑な状態が十分に持続しなければ有効な schema が生成されず、引いては GA が良い近似解を出せない。逆に、遺伝子の乱雑さを何らかの方法で持続できれば GA がより有効に働くと推測される。

3 Crossover の基本コンセプト

最初に解空間が数直線上にあるような関数の最適化における crossover を考える。我々の基本的アプローチは、数直線上の適当な2点を選び、その中点の適応度関数値と前の2点の適応度関数値を比較し、良いものを次の2点とする再帰的分割に基づいている。

解空間を $[0,1]$ の実数数直線に見立て、最初の2点を 0,1 とすると、そこから生成される解は次表のようになる。

class	position	
0	n	$(n=0,1)$
1	$n\cdot\frac{1}{2}$	$(n=1)$
2	$n\cdot\frac{1}{2^2}$	$(n=1,3)$
3	$n\cdot\frac{1}{2^3}$	$(n=1,3,5,7)$
⋮	⋮	⋮
t	$n\cdot\frac{1}{2^t}$	$(n=1,3,\cdots,2^t-1)$

(ただし、position は数直線上の位置、class は何度目の操作によって生成された遺伝子かを表わす。)

そこで、2点から中点を作り出す行程を GA の crossover と見立てることにする。この場合、crossover は以下の性質を有する。

class t_1、class t_2 の遺伝子が crossover すると、新たに生成される遺伝子の position は次で与えられる。(ただし、n_1, n_2 は奇数)

- $t_1 < t_2$ の場合

$$\begin{aligned} position &= (n_1\cdot 2^{-t_1} + n_2\cdot 2^{-t_2})/2 \\ &= 2^{-t_2-1}(n_1\cdot 2^{t_2-t_1} + n_2) \end{aligned}$$

ここで、$(n_1\cdot 2^{t_2-t_1}+n_2)$ は奇数であるので、生成された遺伝子は class (t_2+1) に属する。

- $t_1 = t_2$ の場合

$$position = 2^{-t_1}((n_1+n_2)/2)$$

となり、$(n_1+n_2)/2$ が整数となるので、生成される遺伝子は class t_1 に属する。

以上のことから class が異なる遺伝子同士を crossover させ、より大きな class の遺伝子を生成していけば、重複無しで解空間を探索できることが分かる。

4 バイナリ空間への適用

前節で述べた基本的考え方を、バイナリで符合化された遺伝子に適用するために、遺伝子の属性として cross pointer (以下 CPR と略す。) を導入する。ここではバイナリで符合化された遺伝子を念頭に置いて、class の定義と CPR の説明を行なう。

4.1 class

通常の GA では遺伝子の優位性を population 内でその遺伝子が占める割合という形で表現しているが、これでは遺伝子の重複を許し、世代が進むにしたがって探索が妨げられることになる。そこで我々は、遺伝子の優位性を適応度関数値でのみ表現し、population 内での遺伝子の重複はないものとした。

上記のことを実現するため前節の考えを取り入れ、crossover で生成される遺伝子をその親となる遺伝子の (解空間上におけるなんらかの距離に対する) 中点になるように生成していけば、仮に population が十分にある場合解空間を重複を出さずに網羅できると考えられる。

前節で述べた class の性質より、遺伝子を重複させないような構造を作るには、遺伝子が解空間内の最大 class に到達するまで GA 操作を繰り返せば良いことになる。

しかし、この方法をバイナリ空間へ適用する場合、遺伝子間の距離をハミング距離等で与えるのは良いとしても、その中点が unique

A Study on Search Performance of Genetic Algorithms
Toshio HITAKA, Yoshiaki TAKAI, Yoshiharu SATOH
HOKKAIDO UNIVERSITY

に決まらないという点が問題となる。そこで、今回はハミング距離で中点となる遺伝子の内、one point crossover 一回の操作で生成できるものを採用することにした。

実際の制御は各遺伝子に持たせたCPRという属性を用いて行なう。

4.2 CPR

最初に、遺伝子 A,B（ただし、class は B の方が大きいとする。）を与えた場合、crossover がどのように行なわれるのかを述べる。

各 A,B ともCPRとして a,b を持っているものとする。次に、各CPR から check point 1,2 を決定し、その bit 値が異なる場合は $(a+b)/2$（図中 cp1）を、等しい場合には $b-(a-b)/2$（図中 cp2）を crossover point として採用する。(Figure.2 参照)

一方、新しく生成された遺伝子の CPR は、crossover の際に用いられた crossover point の位置となる。

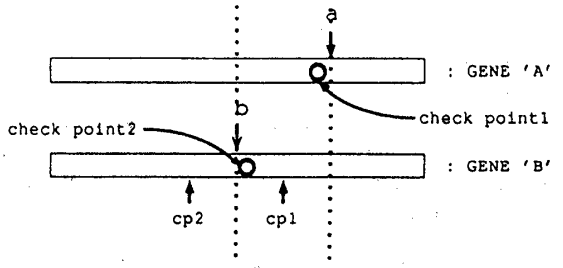

Figure.2：CPR を用いた crossover の方法

続いて初期遺伝子集団の生成法であるが、SGA のようにランダムに遺伝子を選ぶのではなく、最初に遺伝子集団の元となる種 (seed) をランダムに選出し、その遺伝子の各 bit の補数からなる遺伝子とで class0 を構成させ、class1 は class0 同士の crossover により遺伝子の生成を行なう。

その後遺伝子数が、必要な個数を満たすまで class n と class n-1 の crossover を繰り返す。

以下に遺伝子が生成されていく様子を示す。なお、遺伝子中のマーク（▼）は CPR の示す位置である。

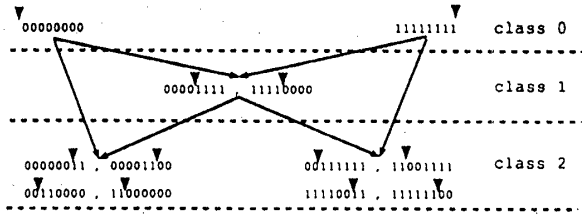

Figure.3：初期遺伝子生成の例

上記の方法で初期遺伝子を生成し、その後の GA 操作においても同様な crossover を用いて遺伝子生成を行なう。ただし、同一の class 属性を持つ遺伝子同士は crossover させないように制限する。

4.3 アルゴリズムの構築

以上述べたことを 1 つのアルゴリズムに組み立てる。

提案した方法で初期遺伝子集団を形成した後、遺伝子が解空間中の最大 class へ到達か、もしくは何らかの収束条件を満たすまで selection と crossover を繰り返す。

selection の方法については、遺伝子の重複を出さないという条件以外には制約はない。

なお、遺伝子が十分に分散しているということと、遺伝子の重複を入れないという観点から mutation 操作は入れないものとする。

ここで、seed 生成から世代を繰り返し、重複が出る直前までの一連の操作を cycle と呼ぶことにする。

seed を変更しつつ cycle を複数回実行し、各 cycle から得られた近似最適解の内、最大の適応度関数値を最終的に得られる解とする。

一連の流れを Figure.4 に示す。

```
repeat
  make 'SEED'
  generate 'GENES' from 'SEED'
  while not MAX class                    ┐
       selection                         │ 1 cycle
       crossover                         │
    end while                            ┘
until stopcriterion
```

Figure.4：提案するアルゴリズム

なお、selection において元の遺伝子とそこから生成された遺伝子の全体に対しての淘汰を行なっているため、元の遺伝子が非常に優勢な場合、絶えず生成遺伝子が排除されて遺伝子集団としては変わらない状態が想定されたので、ループの条件判断に制限回数を設けている。

5 実験結果

今回はナップザック問題を例にとり、SGA との比較実験を行なった。問題の遺伝子は 32bit のバイナリで表現されており、厳密解は２６８３である。

各 cycle 内の、GA の世代毎に 適応度関数値の最大値 の１００回平均をプロットしたものが以下のグラフである。

実線は今回提案した GA、破線は SGA を表す。

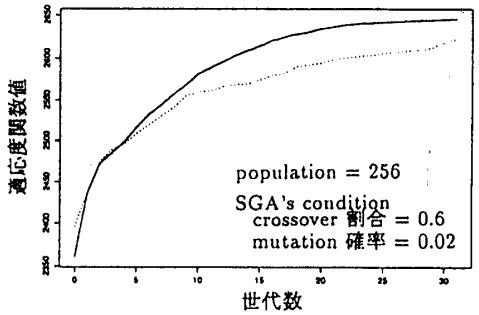

Figure.5：提案する GA と SGA の比較実験

ここでは、1cycle につき 32 世代（最大 class 到達までの最短世代数）となっているが、十分収束状態になるようである。数回実験を行なったが、概ね 2550 以上の解が出ており、SGA との比較の結果、提案した GA は効率向上と言う点で明らかに優位であると思われる。

6 まとめ

遺伝子重複を押さえるための crossover 方法を提案し、その有効性をナップザック問題を例に取って実験的に示した。

seed に対する制御の点に考察の余地があるが、遺伝子重複がないという点は達成されており、さらに各 cycle 終了時の遺伝子状態を見てみた場合、重要な（適応度関数値に対する影響が大きい）bit がほぼ固定しており、遺伝子の絞り込みの点でも今回の GA はかなり有効に働いている。

参考文献

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

[2] K. A. DeJong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems"(PhD Thesis), University of Michigan, 1975.