多峰性問題のDeterministic Crowding法適用における初期世代の交叉と突然変異の影響

姫野雅子[†] 姫野龍太郎[††]

Deterministic Crowding（DC）は，多峰性問題を解くうえで有効な方法である．本論文では，世代により交叉と突然変異の適用がDCの性能にどのような影響を与えるかを調べた．その結果，初期世代に一様交叉や高い突然変異率を適用するとより小さな集団で解を見つけることができた．また，後半世代では解の積み木を積み上げられる二点交叉または一点交叉の適用が有効であった．

The Effect of Crossover and Mutation to DC in Early Generations for Multimodal Function Optimization

Masako Himeno[†] and Ryutaro Himeno[††]

Deterministic Crowding (DC) is suitable for searching for the multiple optimums of multimodal functions. This paper investigates the effect of crossover and mutation in different generations to DC. Multiple solutions can be found at smaller populations by the adapting of uniform crossover or high-rate mutation in early generations. It is found that two-point or one-point crossovers are useful in latter generations which are able to construct the building blocks.

1. はじめに

標準的遺伝的アルゴリズム（sGA）は，単峰性問題の最適値を求めるのに適している[1]．しかし，sGAでは個体群の中で評価値の高い個体の選択と繁殖を繰り返すことで解を求めるため，局所解を多く持つ場合や複数解を持つ多峰性問題を解く場合には困難をきたす[1]．そこで，集団内の多様性を保持するため，ニッチや種の形成を行うなどの各種の遺伝的操作の工夫が施されている．現在ニッチ形成のアルゴリズムは多種類の方法が提案されている[2)~4)]．その1つとして，新個体の置き換え対象を制限することによりニッチを形成し，多様性を保持する方法がある．Mahfoud[3]は，新個体の方が優れている場合に限り，それを生成した親個体のうち新個体に類似した方と置き換えるDeterministic Crowding（DC）と呼ばれる方法を提案した．DCは，複雑な多峰性問題での実験でも有効であったが，複雑な問題になるほどより探索に時間がかかった．また，同種内での個体の置き換えを行うアルゴリズムであるため，高い評価値の種が低い評価値の種を排除する強力な置き換え能力を持っているわけではない．そこで，ある確率で新個体が親よりも評価値が低い場合でも置き換えを行うProbabilistic Crowdingも提案されている[5]．

DCは，多峰性問題を解くうえで有効な方法であり，個体の選択と置き換えという視点では詳細な実験と解析が行われてきたが，交叉や突然変異など他のオペレータを適用したときの解析は行われていない．そこで，ここでは各種の交叉，またはさまざまな突然変異率の適用のみでなく，複数の交叉の適用や実行途中での突然変異率の変更も検討した．実際の動物の進化では遺伝子の進化速度はいつも一定ではなく，進化の初期の方がその後よりも大きいことが分かっている[6]．たとえば，神経系の発生に深く関与する*Pax*遺伝子では，カイメンとその他の動物の分岐以前の進化速度はその後の進化速度の2.4倍であった[7]．これを反映するため，ここでは複数の解を解くことができるDCの性能に，世代によるオペレータの変更がどのような影響を与えるのか調べてみた．

† 桐朋女子高校音楽科
Tohogakuen Music High School
†† 理化学研究所
RIKEN (The Institute of Chemical and Physical Research)

2. 方　　法

2.1 Deterministis Crowding（DC）

今回の解析には Deterministic Crowding（DC）法を使用した．DC は，集団内の多様性を保持する目的で提案された方法である．まず，集団を 2 個体ずつランダムに組み合わせ，その組において交配を行う．交叉や突然変異などのオペレータを適用した後，置き換えを行う．親子 4 個体において，新個体と親個体で似たものを組にし，新個体の方が親個体より評価値が高い場合のみ親個体と入れ替える．集団内の全個体が繁殖の対象になることと，似通った個体を制限することにより集団内の多様性を保持することができる．また，DC では新個体の評価値が親個体のそれより良い場合のみ置き換えを行うため，一度獲得した評価値の高い個体がなくなることはない．

2.2 交叉と突然変異

交叉は代表的な遺伝的オペレータである．交叉には適応度の高いスキーマを組み換える能力があり，そのため独特な探索を行うことができ有効である．また，Jones[8] は交叉には探索空間を大きく移動する「マクロ的な突然変異」の役割もあることを実験により確かめている．よく活用される交叉には一点交叉，二点交叉，一様交叉があげられるが，Rana[9] はこれらの交叉を適用した場合の新個体と親個体の違いを理論と実験から比較している．その結果，sGA で適用した場合一様交叉の方が一点交叉，二点交叉よりも明らかに親子の距離は大きいことが示されている．また，交叉の探索空間の移動能力は初期世代が一番大きく，その後急激に減少することも示されている[10]．どの交叉を使用すればよいかは，GA の他の細部に複雑に依存し，問題の難易度，集団数，コード化によりまちまちである．GA の施行中に適した交叉を見つけ出す試みもあり，Spears[11] は遺伝子の中に交叉を決めるビットを用意し，二点交叉か一様交叉を進化させた．

突然変異は一般的に集団の多様性を保持する働きをするバックグラウンド的なオペレータと考えられていた．しかし，交叉同様に新しい解の発見や発見した解の活用に重要な貢献をしていることが分かってきている[10),12]．Wu ら[12] は，交叉と突然変異の役割を比較したところ，両方とも探索時に重要な役割をするが，突然変異は交叉より新しい解の積み木の構築や発見した解の破壊という点でより強力であり，交叉は解の積み木を次世代に遺伝させる役割がより大きいと述べている．また，使用する突然変異率も問題解決に大きく作用する．ただし，最適な突然変異率は取り扱う問題に依存するので，探索に応じて突然変異率を変化させる研究もある．Bäck[13] は，最適な突然変異率を計算したところ，評価値の上昇とともに突然変異率が下がる傾向があったことを述べている．また，このような突然変異率の変動を適用した方が，一定の突然変異率を使用するよりも有効な場合があることを示した．

多峰性問題を解くアルゴリズムの多くは，一点交叉を使用している[3),4)]．ここでは探索空間の移動距離の大きい一様交叉と，比較的移動距離の小さい二点交叉と一点交叉を取り上げ，これらの交叉を組み合わせることで DC の性能にどのような影響があるかを調べた．また，DC 実行中の突然変異率の変更による影響も試みた．

3. 実験と結果

3.1 テスト関数

$$F1 : \sin^6(5\pi x) \qquad (0 \leqq x \leqq 1)$$

$$F2 : \prod_{k=1}^{n} \sin^6(a_k \pi x_k) \qquad (0 \leqq x_k \leqq 1)$$

F1 関数は等間隔に 5 つのピークを持つ簡単な 1 変数関数である．多峰性問題の解析によく使用されている[2),3),5)]．この関数は遺伝子長 30 ビットで実験を行う場合が多い．そこで今回は 20 ビット（400 世代まで），30 ビット（500 世代まで），40 ビット（600 世代まで）の 3 通りの遺伝子長で実験を行った．

F2 は多変数関数である．n=2（F2[n=2]）と n=3（F2[n=3]）で実験を行った．F2[n=2] では a_1=2, a_2=4 のパラメータを用い遺伝子長 40 ビット（x_1，x_2 ともに 20 ビット）で実験した．8 つのピークを持つ関数である（**図 1**）．F2[n=3] では a_1=2，a_2=4，a_3=3 のパラメータを用い，遺伝子長 60 ビット（x_1，x_2，x_3 ともに 20 ビット）で実験した．24 個の最適解を持つ．F2 関数は 500 世代まで行った．

$$F3 : \frac{1}{1 + |z^6 - 1|} \qquad z = x + iy$$
$$(-2 \leqq x, y \leqq 2)$$

F3 関数の最大値は 1 の 6 乗根の複素数のときである．この関数は複素平面上 (0,0) を中心として高さ 0.5 の平面の周囲に 6 つの鋭いピークを持っている[4)]．x と y は実数値で，両方とも 20 ビットでコードし計 40 ビットの遺伝子長を用いた．800 世代まで行った．

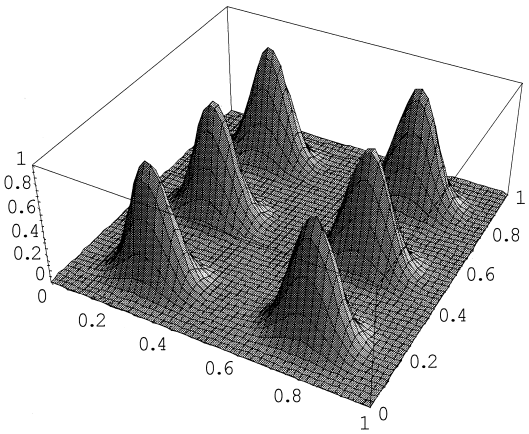


図 1 F2[n=2] 関数
Fig. 1 F2[n=2] function.

$$F4 : \sum_{i=0}^{4} \mathbf{u}\left(\sum_{j=0}^{5} x_{6i+j}\right)$$

$$\mathbf{u}(x) = \begin{cases} 1.0 & x = 0, 6 \\ 0 & x = 1, 5 \\ 0.360384 & x = 2, 4 \\ 0.640576 & x = 3 \end{cases}$$

F4 は強力な騙し問題を含む多峰性関数である[14]．6 ビットを 1 ユニットとし 5 ユニット連続したもので，遺伝子長は 30 ビットである．関数 $u\ (x)$ は，1 単位中の 1 の総数 x に対し上のような値が与えられる．x が 0 と 6 で最高値 1.0，x が 3 で局所解 0.640576 をとる．この関数は 32 の最適解と多数の局所解を持つ．400 世代まで行った．

F1 または F3 の関数を扱っている多くの場合，バイナリコードでコーディングしている．比較検討のため，すべての関数の遺伝子コーディングをバイナリコードで行った．また，それぞれのケースで同じ 50 種類の初期集団を使用して実験を行った．

3.2 F1 関 数

3.2.1 交叉の適用

交叉の影響を見るため突然変異は使用しなかった．**表 1** は遺伝子長 30 ビットで一点交叉（1X），二点交叉（2X），一様交叉（UX）の 3 種類の交叉を適用した結果である．DC 実行中に 5 つすべての最適解を求めた場合を成功とし，表には 50 回の試行中の成功した試行回数と，括弧内にその場合の平均世代数が示されている．また表中の Pc は交叉率を表している．それぞれの交叉を単独で適用した場合の発見頻度は二点交叉が一番良かったが，発見速度は一点交叉が最も良かった．交叉率は 0.8 より 1.0 の方が解の発見頻度，または発見速度に良い傾向が見られた．以下の実験結果は，特に断りがない場合，どの交叉も交叉率 1.0 を用いたものである．

表 1 交叉の影響（遺伝子長 30 ビット）
Table 1 Effect of crossover (gene length of 30 bits).

	70 集団数	140 集団数	200 集団数
1X(Pc=1.0)	0	1(177.0)	4(151.3)
1X(Pc=0.8)	0	1(151.0)	1(142.0)
2X(Pc=1.0)	0	13(224.1)	24(221.3)
2X(Pc=0.8)	0	10(298.5)	24(278.0)
UX(Pc=1.0)	0	0	0
UX(Pc=0.8)	0	0	0

表 2 交叉の組合せ
Table 2 Combination of crossover.

	70 集団数	140 集団数	200 集団数
(0-10)UX+1X *	0	6(189.7)	16(159.6)
(0-30)UX+1X	1(172.0)	15(175.8)	27(166.3)
(0-100)UX+1X	4(368.2)	20(245.9)	34(223.4)
(0-10)UX+2X	2(219.0)	13(224.8)	29(196.7)
(0-30)UX+2X	1(291.0)	26(262.5)	36(214.6)
(0-100)UX+2X	3(321.3)	25(279.0)	41(260.7)
(0-30)2X+1X	0	5(124.8)	13(144.5)
(0-30)1X+2X	0	3(192.0)	3(266.7)

*(0-10)UX+1X：10 世代まで一様交叉，その後一点交叉

次に 3 種類の交叉を組み合わせて実験を行った．交叉の組合せは，初期世代に一様交叉その後一点交叉または二点交叉，初期世代に一点交叉その後二点交叉，初期世代に二点交叉その後一点交叉の 4 通りである．初期世代に一様交叉を適用した場合はその適用世代（g）を 10 世代まで（g=10），30 世代まで（g=30），100 世代まで（g=100）の 3 種類行った（**表 2**）．表中では，たとえば初期世代の 10 世代まで一様交叉でその後一点交叉を適用したものを *(0-10)UX+1X* のように表している．初期世代に一様交叉を適用し後半に一点交叉を行ったもの（(0-g)UX+1X）は全世代一点交叉（1X）の場合よりもより小集団で解を発見でき，また同数の集団数では解の発見頻度が高かった．同様の傾向が，全世代二点交叉（2X）と比較したときの初期世代に一様交叉を適用した場合（(0-g)UX+2X）に見られた．一様交叉を適用する世代を長くすると解の発見頻度は高くなるが，発見速度が遅くなる傾向が見られた．また，後半に二点交叉を適用した方が一点交叉の適用より解の発見頻度は高かったが，発見まで時間がかかった．

30 世代までの初期世代に二点交叉を適用し後半一点交叉を行ったもの（(0-30)2X+1X）は，全世代一点交叉（1X）の場合よりは発見頻度が高かったが，初期世代に一様交叉を適用した場合（(0-30)UX+1X）ほ

表 3 一様交叉の適用世代の違い
Table 3 Effect of the generations adapting uniform crossover.

	70 集団数	140 集団数	200 集団数
(0-30)UX+1X(470)*	1(172.0)	15(175.8)	27(166.3)
(31-60)UX+1X(500)	0	2(319.5)	3(156.0)
(31-60)UX+1X(530)	0	2(319.5)	3(156.0)
(0-30)UX+2X(470)	1(291.0)	25(253.3)	35(207.1)
(31-60)UX+2X(500)	0	15(266.6)	25(240.2)
(31-60)UX+2X(530)	0	15(266.6)	25(240.2)

*470 世代まで行う

表 4 交叉率の影響（200 集団数）
Table 4 Effect of the crossover provability (200 populations).

	1X Pc=1.0	1X Pc=0.8	1X Pc=0.6
UX Pc=1.0	27(166.3)*	25(173.6)	17(192.4)
UX Pc=0.8	24(157.2)	19(186.3)	15(202.9)
UX Pc=0.6	23(160.3)	15(163.3)	14(208.4)
	2X Pc=1.0	2X Pc=0.8	2X Pc=0.6
UX Pc=1.0	36(214.6)	36(252.2)	29(298.5)
UX Pc=0.8	34(210.4)	35(260.5)	26(304.9)
UX Pc=0.6	31(222.7)	30(286.6)	25(301.6)

*30 世代まで一様交叉 (Pc=1.0)，後半一点交叉 (Pc=1.0)

どの改善は見られなかった．一方初期世代に一点交叉を適用した場合（(0-30)1X+2X）は，全世代二点交叉（2X）を行った場合より解の発見頻度が低下した．

次に一様交叉を適用する時期を中間世代にしてその効果を調べた．その結果が**表 3** である．一様交叉を適用する世代は，発見頻度の改善が見られかつ発見速度が一様交叉を適用しない場合とほぼ同様の結果が得られる 30 世代とした．31 世代から 60 世代まで一様交叉を適用する（(31-60)UX）と，30 世代まで一様交叉を適用した場合（(0-30)UX）より一点交叉を適用する世代が 30 世代短くなる．そこでその影響をなくすため，(0-30)UX を 470 世代まで実行する場合と (31-60)UX を 530 世代まで実行する場合も行った．中間世代に一様交叉を適用した場合には初期世代に適用したほどの効果は得られなかった．

初期世代の 30 世代まで一様交叉，後半に一点交叉または二点交叉を適用する場合の交叉率の影響を調べた結果が**表 4** である．どの組合せの交叉でも，交叉率を下げると解の発見頻度または発見速度が低下した．

遺伝子長 20 ビットと 40 ビットでの結果が**表 5** である．交叉を単独で用いた場合，解の発見頻度では二点交叉が，発見速度では一点交叉が最も良かった．一点交叉，二点交叉を使用する場合，初期世代に一様交叉を適用する（(0-30)UX+1X,(0-30)UX+2X）と解の発見頻度が良くなったが，中間世代で適用する（(31-60)UX+1X,(31-60)UX+2X）と初期世代で適用したほどの改善は見られなかった．以上 30 ビットと同様の傾向が見られ，ビット数による違いはなかった．

表 5 遺伝子長 20 ビットと 40 ビットでの結果
Table 5 Experimental results for gene length of 20 bits and 40 bits.

20 bits	40 集団数	100 集団数
1X	1(116.0)	18(120.8)
2X	3(216.0)	35(156.5)
UX	1(380.0)	15(268.3)
(0-30)UX+1X	7(151.2)	42(102.3)
(31-60)UX+1X	3(129.0)	22(118.8)
(0-30)2X+1X	4(148.2)	31(114.6)
(0-30)UX+2X	10(192.7)	41(153.7)
(31-60)UX+2X	5(213.7)	36(171.9)
(0-30)1X+2X	1(199.0)	19(138.3)
40 bits	300 集団数	700 集団数
1X	1(180.0)	7(253.0)
2X	3(408.0)	19(322.2)
UX	0	0
(0-30)UX+1X	4(326.5)	27(209.6)
(31-60)UX+1X	0	2(198.5)
(0-30)2X+1X	2(285.7)	21(210.7)
(0-30)UX+2X	6(404.4)	36(333.1)
(31-60)UX+2X	3(399.7)	25(346.8)
(0-30)1X+2X	1(320.0)	6(322.7)

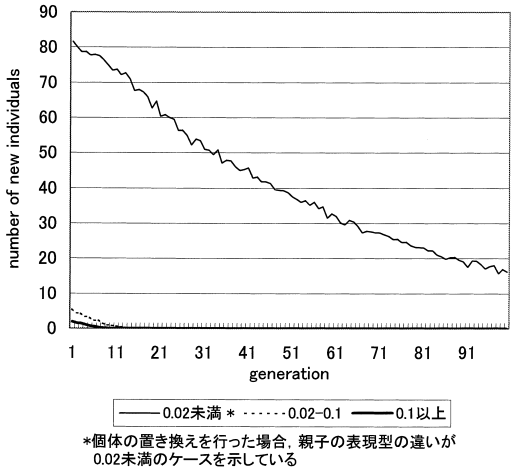


図 2 全世代一点交叉適用時の置き換えた新個体と親個体の表現型距離
Fig. 2 Difference of phenotype between parents and replaced offspring in adapting one-point crossover.

遺伝子から求められる x の値を表現型とし，各世代で置き換えた新個体と親個体の表現型の違いを調べてみた．その結果が**図 2**，**3**，**4** である．遺伝子長 30 ビット，200 集団数を使った．親との表現型の違いを 3 ランクに分け，そのランクに入る新個体の個体数を図にした．x_{parent} を親の表現型，x_{child} を子供の表現型としたとき，親子間の表現型の距離 d は次のよう

表 6 突然変異率と適用する世代の関係
Table 6 Relationship between mutation rate and adapting generations.

突然変異率	0.01	0.05	0.1	0.3	0.5
全世代に適用	0	3(447.3)	0	0	0
(0-30)mu +1X*	10(150.3)**	19(164.9)	30(177.7)	28(161.1)	29(153.4)
(0-50)mu +1X	18(234.6)	29(226.9)	41(222.2)	45(212.4)	33(211.5)
(0-30)mu +2X***	29(210.1)	33(225.8)	36(220.7)	44(207.9)	40(220.7)
(0-50)mu +2X	27(285.0)	40(257.1)	45(258.0)	49(244.9)	49(257.6)

*(0-30)mu +1X：30 世代まで突然変異を適用しその後一点交叉を行う．
** 30 世代まで変異率 0.01 の突然変異を適用しその後一点交叉を行う．
***(0-30)mu +2X：30 世代まで突然変異を適用しその後二点交叉を行う．

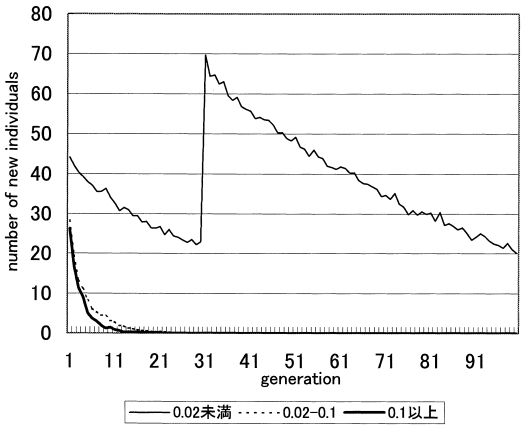


図 3 30 世代までの一様交叉適用時の置き換えた新個体と親個体の表現型距離
Fig. 3 Difference of phenotype between parents and replaced offspring in adapting uniform crossover at early 30 generations.

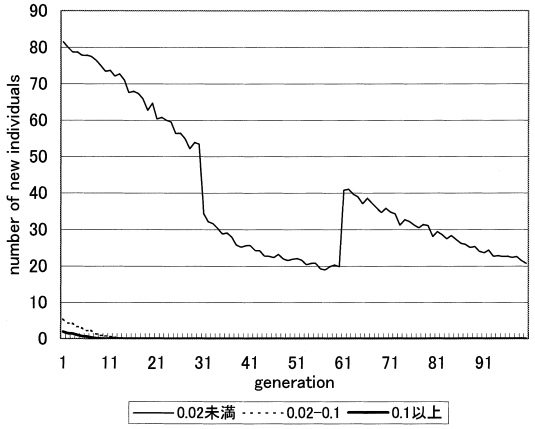


図 4 31 から 60 世代までの一様交叉の適用時の置き換えた新個体と親個体の表現型距離
Fig. 4 Difference of phenotype between parents and replaced offspring in adapting uniform crossover from 31 generation to 60 generation.

に表される．

$$d = |x_{parent} - x_{child}|$$

一点交叉のみのもの（図 2）と初期世代に一様交叉を適用したもの（図 3）とを比較すると，一点交叉のみのものは親との差がほとんど 0.02 未満の個体であるが，一様交叉を適用したものは一様交叉を行っている世代で差が 0.1 以上の個体もあった．このことから一様交叉では一点交叉に比べて置き換えをした新個体は親個体との表現型の違いが大きいことが分かる．これは一様交叉を行うことでより広い範囲の探索を行い，集団内の多様性を維持できることを示している．良い効果が見られなかった中間世代で一様交叉を適用した場合は，図 4 のように一様交叉を行っている間置き換わった個体が極端に少ない．この時期に一様交叉を行っても親個体よりも良い個体が得られず，結果的に DC の性能をそれほど上げることができなかったと考えられる．同様の結果が二点交叉を後半に適用する場合にも見られた．

以上より DC においては，まだ解の候補が見つかっていない初期世代により広い範囲の探索をすることが後の検索に良い影響を与えるようである．一方，解の候補を発見している世代での一様交叉はその解のスキーマを破壊する方向に働くのであろう．このことをさらに確認するため，初期世代と中間世代に高い突然変異率を適用しその影響を観察した．

3.2.2 突然変異の適用

一様交叉の代わりに突然変異を適用した．一様交叉の結果と比較するため，突然変異を適用している世代では交叉を行わなかった．5 種類の変異率の突然変異を，適用する世代を変えて実験した結果が**表 6** である．遺伝子長 30 ビット，集団数 200 個体で行った．突然変異のみの適用では，500 世代までには解を発見できたのは変異率 0.05 の場合のみであったが，この場合も発見までに時間がかかった．次に，初期世代に突然変異を適用しその後一点交叉または二点交叉を試みた．表中では，30 世代まで突然変異を行いその後一点交叉を適応した場合を，*(0-30)mu+1X* と表している．突然変異率が高い 0.1，0.3，0.5 を適用した

表 7 突然変異率と集団数の関係
Table 7 Relationship between mutation rate and generation size.

	70 集団数	140 集団数
(0-30)mu0.01+1X	0	2(139.0)
(0-30)mu0.3+1X	1(435.0)	14(193.6)
(31-60)mu0.3+1X	0	3(146.3)
(0-30)mu0.01+2X	1(429.0)	20(270.8)
(0-30)mu0.3+2X	5(243.8)	28(220.1)
(31-60)mu0.3+2X	1(304.0)	20(271.5)

方が低い突然変異率を適用した場合より解の発見頻度が高かった．また，突然変異を適用する世代を増やした方が解の発見頻度が高くなるが，発見まで時間がかかった．**表 7** は，30 世代までの初期世代（0-30）と 31 から 60 世代の中間世代（31-60）に突然変異を適用した場合について集団数を変えて実験した結果である．表中 (0-30)mu0.01+1X は，30 世代まで変異率 0.01 の突然変異を適用した後一点交叉を行ったことを表している．後半に一点交叉または二点交叉を行った場合，初期世代により高い突然変異率を適用した方（(0-30)mu0.3+1X,(0-30)mu0.3+2X）がより小さい集団で解を求めることができた．また，同じ 30 世代の高突然変異率の適用でも，初期世代を避け 31 世代から 60 世代の適用（(31-60)mu0.3+1X,(31-60)mu0.3+2X）は解の発見頻度を低下させた．この傾向は，遺伝子長 20 ビットと 40 ビットでも見られた（**表 8**）．

一点交叉，二点交叉ともに初期世代（0-30）と中間世代（31-60）に変異率 0.3 の突然変異を適用した場合の各世代での置き換えをした新個体と親個体の表現型の違いを調べたところ，初期世代に適用した場合は，親との違いが 0.1 以上の個体が 0.02 未満の個体より多かった．一様交叉の適用と同様に広い範囲の探索が行われていることが分かった．一方，中間世代での適用は置き換わる個体数が著しく少なかった．このため DC の性能の改善には至らなかったと考えられる．

3.2.3 スキーマの構築

最終的な解を求めるためには，世代の後半で発見したスキーマの構築を行うオペレータを働かせる必要がある．**表 9** は 30 世代まで変異率 0.3 の突然変異を適用した後それぞれのオペレータを適用した結果である．遺伝子長 30 ビット，集団数 200 個体を用いた．後半世代では，突然変異より二点交叉または一点交叉を用いた方が解の発見率は高かった．また，二点交叉の方が一点交叉より発見頻度は高かったが発見速度は遅かった．

表 8 遺伝子長 20 ビットと 40 ビットでの結果
Table 8 Experimental results for gene length of 20 bits and 40 bits.

20 bits	40 集団数	100 集団数
(0-30)mu0.01+1X	0	22(128.1)
(0-30)mu0.3+1X	19(134.1)	46(96.6)
(31-60)mu0.3+1X	4(129.0)	39(112.5)
(0-30)mu0.01+2X	8(248.3)	41(172.3)
(0-30)mu0.3+2X	20(207.2)	47(132.5)
(31-60)mu0.3+2X	7(202.1)	41(160.8)
40 bits	300 集団数	700 集団数
(0-30)mu0.01+1X	3(208.3)	11(280.8)
(0-30)mu0.3+1X	5(258.0)	25(215.3)
(31-60)mu0.3+1X	0	10(251.6)
(0-30)mu0.01+2X	3(390.7)	21(380.2)
(0-30)mu0.3+2X	15(385.4)	37(321.6)
(31-60)mu0.3+2X	3(424.7)	28(373.3)

表 9 後半世代でのオペレータの効果
Table 9 Effects of operators in latter generations.

交叉率	1.0	0.8	0.6
1X	28(161.5)	30(187.0)	27(212.2)
2X	44(207.9)	38(239.4)	41(304.5)
UX	6(453.5)	0	0
変異率	0.1	0.05	0.01
突然変異	0	5(417.2)	2(464.0)

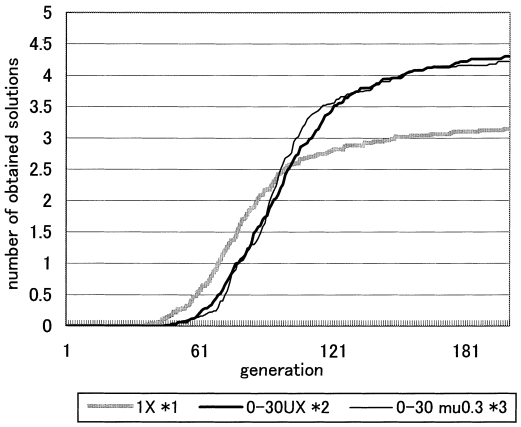


図 5 発見した解の個数
Fig. 5 Number of obtained solutions.

3.2.4 解の発見速度

図 5 は，全世代一点交叉，30 世代まで一様交叉後一点交叉（(0-30)UX+1X），30 世代まで 0.3 突然変異後一点交叉（(0-30)mu0.3+1X）を適用した場合の各世代における発見した解の個数（50 試行の平均値）をグラフにしたものである．この実験では遺伝子長 30 ビット，集団数 200 個体を用いた．一点交叉のみを用

表 10 F2 関数の結果
Table 10 Experimental results for F2 function.

F2[n=2]	150 集団数	300 集団数
1X	2(177.0)	15(158.4)
2X	6(257.7)	22(210.8)
UX	0	0
(0-30)UX+1X	7(185.7))	28(148.5)
(31-60)UX+1X	3(191.7)	18(213.7)
(0-30)2X+1X	1(135.0)	20(143.2)
(0-30)mu0.3+1X	6(227.5)	36(178.8)
(31-60)mu0.3+1X	3(183.0)	20(206.7)
(0-30)UX+2X	12(220.3)	36(180.2)
(31-60)UX+2X	4(208.8)	16(202.7)
(0-30)1X+2X	4(215.0)	14(227.2)
(0-30)mu0.3+2X	18(247.2)	45(198.1)
(31-60)mu0.3+2X	7(301.4)	22(217.8)
F2[n=3]	700 集団数	1500 集団数
1X	0	4(174.0)
2X	5(322.0)	22(222.9)
UX	0	0
(0-30)UX+1X	3(211.0)	13(184.1)
(31-60)UX+1X	0	4(177.0)
(0-30)2X+1X	2(231.0)	10(177.0)
(0-30)mu0.1+1X	6(190.2)	21(170.9)
(31-60)mu0.1+1X	0	9(178.0)
(0-30)UX+2X	8(312.5)	27(260.0)
(31-60)UX+2X	1(324.0)	15(251.8)
(0-30)1X+2X	1(234.0)	12(244.1)
(0-30)mu0.1+2X	9(339.6)	42(238.5)
(31-60)mu0.1+2X	3(219.3)	23(255.7)

いた場合は，早い世代で解を見つけ始めるが頭打ちになる世代も早い．それに対し，初期世代で一様交叉や高率の突然変異を適用した方は解を見つけ始める時期は遅れるものの，いったん発見し始めると急速にその数を増やしていく．最終的には，一点交叉のみの場合より多くの解を見つけることができた．このことから，一点交叉のみの場合は，探索が十分に行われないうちに集団の多様性が減少してしまい多数の解を見つけることができないことが分かる．後半に二点交叉を用いた場合も同様の結果が得られた．

3.3 その他の関数

他の関数で F1 関数と同様の実験を行った（**表 10**, **11**, **12**）．結果は以下のとおりである．

(1) 交叉を単独で用いた場合二点交叉が最も解の発見頻度が高かった．F2 関数では，解発見までの時間は二点交叉より一点交叉の方が速かったが，F3 と F4 関数ではほぼ同じスピードで発見できた．一様交叉ではどの関数でも解を求めることはできなかった．

(2) 2 種類の交叉を組み合わせたところ，F1 関数同様に一点交叉，二点交叉ともに初期世代に一様交叉を適用すると解の発見頻度が高くなった．一点交叉，二点交叉単独での発見速度とほぼ同じで，かつ解の発見頻度を上げることができる一様交叉の適用世代は関数によりまちまちで，F2 関数では 30 世代，F3 関数では 50 世代，F4 世代では 20 世代であった．

(3) 初期世代に二点交叉その後一点交叉を行ったものは，一点交叉のみの場合より解の発見頻度は良くなり，中には F2[n=3] や F4 関数のように初期世代に一様交叉を適用した場合とほぼ同等の改善が見られたものもあった．逆に初期世代に一点交叉その後二点交叉を行ったものは二点交叉のみを適用した場合より解の発見頻度は悪くなった．

(4) 初期世代に高率の突然変異（mu0.1 または mu0.3）を適用すると，一点交叉または二点交叉のみの場合より解の発見頻度が良くなった．最も改善が見られた初期世代に適用する突然変異率は関数により多少異なり F2[n=2] と F4

表 11 F3 関数の結果
Table 11 Experimental results for F3 function.

	200 集団数	400 集団数
1X	0	4(301.2)
2X	13(440.0)	38(288.3)
UX	0	0
(0-50)UX+1X	5(276.4)	25(267.7)
(51-100)UX+1X	1(700.0)	1(736.0)
(0-50)2X+1X	2(267.0)	20(228.8)
(0-50)mu0.1+1X	8(423.9)	23(286.9)
(51-100)mu0.1+1X	2(471.0)	15(487.9)
(0-50)UX+2X	21(373.7)	44(254.1)
(51-100)UX+2X	13(428.2)	35(336.2)
(0-50)1X+2X	3(330.7)	11(395.4)
(0-50)mu0.1+2X	35(348.5)	48(267.3)
(51-100)mu0.1+2X	19(450.8)	43(334.7)

表 12 F4 関数の結果
Table 12 Experimental results for F4 function.

	200 集団数	400 集団数
1X	0	11(102.0)
2X	1(89.0)	29(83.9)
UX	0	0
(0-20)UX+1X	1(95.0)	27(80.5)
(21-40)UX+1X	0	10(113.9)
(0-20)2X+1X	1(74.0)	23(74.9)
(0-20)mu0.3+1X	5(102.2)	35(81.8)
(21-40)mu0.3+1X	0	10(101.4)
(0-20)UX+2X	6(97.2)	39(89.6)
(21-40)UX+2X	2(116.5)	29(99.4)
(0-20)1X+2X	1(99.5)	26(77.7)
(0-20)mu0.3+2X	14(95.3)	47(77.7)
(21-40)mu0.3+2X	1(93.0)	37(99.8)

関数では 0.3，F2[n=3] と F3 関数では 0.1 であった．

(5) 一様交叉または高率の突然変異を中間世代に適用するとどの関数でも初期世代での適用ほどの改善が見られなかった．

4. 考　　察

交叉には発見した解の積み木を積み上げる役割と検索範囲を広げる「マクロ的突然変異」の役割を持っている[8]．そして，一点交叉，二点交叉，一様交叉の順で探索範囲を広げることができ，一様交叉は一点交叉または二点交叉より探索範囲拡大に有効であることが報告されている[9]．この一様交叉の持つ能力が DC の性能を向上させたと考えられる．初期世代に二点交叉後一点交叉を適用した場合は，初期世代に一様交叉を適用したほどではないが一点交叉のみの場合より発見頻度が上昇した．これは二点交叉の方が一点交叉より探索範囲を広げることができたためと考えられる．一方，初期世代に一点交叉その後二点交叉を適用した場合は二点交叉のみと比べ解の発見頻度が悪くなった．以上より，初期世代により探索範囲を拡大できるオペレータを適用するのが DC の性能向上に有効であると考えられる．

高率の突然変異の適用の方が一様交叉の適用より若干有効であったが，これは突然変異が持つ新しい解の構築能力が交叉より優れていることによると考えられる．突然変異は集団中の多様性の有無に限らず任意の範囲を検索することができる[10]．一方，一様交叉の探索範囲の拡大は集団に多様性が保持されている初期世代に限られ，世代が進むにつれ急速に低下する[9]．そのため，突然変異はより広い範囲の探索を行う点では交叉より有利である．

初期世代での高率の突然変異または一様交叉の適用は DC の性能向上に有利であったが，これがいつも成功するとは限らない．DC においては新個体が親よりも適合度が高い場合のみ置き換えを行うため，高率の突然変異または一様交叉の適用で極端に適合度の低い個体が生まれても置き換えの対象にならない．そのため DC は，解のスキーマの破壊により適合度の低い個体が生まれる可能性が高い方法も思い切って行える特徴を持ち合わせている．

一方後半は，ランダムに探索する突然変異の適用よりは，発見できた解の積み木を積み上げられるような二点交叉または一点交叉を適用する方が有効であった．これは交叉の方が，発見した解の積み木を効率良く積み上げることができる[10] ためであると考えられる．

5. 結　　論

4 種類の関数の解析より，交叉を単独で用いる場合一点交叉，二点交叉，一様交叉の 3 種類の交叉のうち二点交叉が一番解の発見頻度が高かった．一点交叉では発見頻度が二点交叉より低くなるが，関数によっては発見速度は速かった．一方，一様交叉では解の発見は難しかった．一点交叉または二点交叉を用いる場合，初期世代に一様交叉または高率の突然変異を適用すると，より小さな集団で複数すべての解を求めることができ，DC の性能を向上させた．解の候補が見つかり始めた中間世代での適用では初期世代ほどの有効性が確認できなかったことから，まだ解の候補を見つけていない初期の時期により広い範囲を探索するのが DC の性能向上に有効であると考えられる．一様交叉や高率の突然変異を適用した方が親子の表現型の違いが大きかったことから，これらオペレータの適用により，より広い範囲の検索が行われたことを示している．高率の突然変異の適用の方が一様交叉の適用より良い結果が得られた．これは突然変異が集団中の多様性の有無によらず任意の範囲を検索し，新しい解の構築能力が交叉より優れていたためと考えられる．一方後半は，ランダムに探索する突然変異の適用よりは，発見できた解の積み木を積み上げられるような二点交叉または一点交叉を適用する方が有効であった．

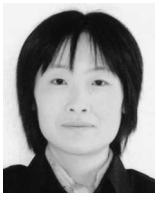
謝辞　この研究では，橋本直樹氏に貴重なアドバイスをいただきました．ここに記して感謝します．

参 考 文 献

1) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, MA (1989).
2) Goldberg, D.E. and Richardson, J.: Genetic Algorithms with sharing for Multimodal Function Optimization, *Proc. 2nd International Conference on Genetic Algorithms*, pp.41–49 (1987).
3) Mahfoud, S.W.: A Comparison of Parallel and Sequential Niching Methods, *Proc. 6th International Conference on Genetic Algorithms*, pp.136–143 (1995).
4) Pétrowski, A.: A New Selection Operator Dedicated to Speciation, *Proc. 7th International Conference on Genetic Algorithms*, pp.144–151 (1997).
5) Mengshoel, O.J. and Goldberg, D.E.: Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement, *Proc. Genetic and Evolutionary Computation Cnference*, pp.409–

416 (1999).
6) 菅　裕，星山大介，宮田　隆：カンブリア爆発と遺伝子多様化，蛋白質 核酸 酵素，Vol.44, No.3, pp.207–216 (1999).
7) Hoshiyama, D., Suga, H., Iwabe, N., Koyanagi, M., Nikoh, N., Knma, K., Matsuda, F., Honjo, T. and Miyata, T.: Sponge *Pax* cDNA Related to *Pax-2/5/8* and Ancient Gene Duplications in the *Pax* Family, *Journal of Molecular Evolution*, Vol.47, pp.640–648 (1998).
8) Jones, T.: Crossover, Macromutation and Population-based Search, *Proc. 6th International Conference on Genetic Algorithms*, pp.73–80 (1995).
9) Rana, S.: The Distributional Biases of Crossover Operators, *Proc. Genetic and Evolutionary Computation Cnference*, pp.549–556 (1999).
10) Spears, W.M.: Crossover or Mutation?, *Proc. Foundations of Genetic Algorithms Workshop 2*, pp.221–237 (1993).
11) Spears, W.M.: Adapting Crossover in Evolutionary Algorithms, *Proc. 4th Annual Evolutionary Programming Conference*, pp.367–384 (1995).
12) Wu, A.S., Lindsay, R.K. and Riolo, R.L.: Empirical Observations on the Roles of Crossover and Mutation, *Proc. 7th International Conference on Genetic Algorithms*, pp.362–369 (1997).
13) Bäck, T.: Optimal Mutation Rates in Genetic Search, *Proc. 5th International Conference on Genetic Algorithms*, pp.2–8 (1993).
14) Goldberg, D.E., Deb, K. and Horn, J.: Massive Multimodality, Deception and Genetic Algorithms, *Parallel Problem Solving from Nature 2*, Manner, R. and Manderick, B. (Eds.), pp.37–46 (1992).

(平成 12 年 8 月 17 日受付)
(平成 13 年 11 月 14 日採録)



姫野　雅子（正会員）

昭和 37 年生．昭和 61 年お茶の水女子大学大学院理学研究科生物学専攻修士課程修了．桐朋女子高校音楽科勤務．桐朋学園大学音楽学部講師．遺伝的アルゴリズム，自然史に興味を持つ．人工知能学会，日本植物学会各会員．

姫野龍太郎（正会員）

昭和 30 年生．昭和 54 年京都大学大学院工学研究科電気工学専攻修士課程修了．工学博士．現在，理化学研究所・情報基盤研究部情報環境室室長．コンピュータ・シミュレーションの研究に従事．日本機械学会，日本可視化情報学会等会員．