形質遺伝を重視した突然変異の提案とその有効性

Proposal of the Character-Preserving Mutation and its Effectiveness

樋口 光明*

Abstract

The author has conducted research mainly into the use of genetic algorithm as a problem solving method applied to scheduling and combinational problem. A main feature of both kinds of problems is that their research objects do not permit multiple selection, therefore genetic manipulation has been carried out using order expression. However, when "crossover" and "mutation" are carried out in the order expression this breaks the good gene sequence from the previous generation.

Therefore "crossover" and "mutation" do not necessarily produce a very good individual.

In another paper a different method called the "sub tour crossover" method was proposed. This method makes it possible to preserve the genotype as the genetic expression, therefore allowing the good genes from the previous generation to be kept.

I wanted to make a mutation which emphasized characteristic heredity and devised and experimented with a new "swap type mutation".

I tested this new genetic manipulation method and announced it at the JCKBSE'2000, but there was no proof of sufficient effectiveness at that time.

However, after systematic experimentation with this method, the effectiveness of this technique was repeatedly confirmed and will be reported.

1。はじめに

GAを用いた、スケジューリング問題適用の試みは興味ある研究分野である。以前、「ペナルティ格差」という新しい概念を導入することによって、合成樹脂工場のスケジューリング

*HIGUCHI, Mitsuaki［情報システム学科］

を試み、現在実施しているエキスパートシステムより良い結果が得られた[3][4]。しかしこの研究対象は重複選択を許さないという大きな性質を持っている。そのため、個体表現型を順序表現にして遺伝子操作を行ってきた。しかし、順序表現で「交叉」及び「突然変異」を行うと、前の世代の良い遺伝子が原形を留めなくなるので、「交叉」や「突然変異」が必ずしも優秀な個体を生み出すとは限らないのではないかと考えた。一方、前の世代の良い遺伝子も残る可能性があり、遺伝子表現も遺伝子型（G-type）のまま交叉が出来るサブツアー交叉が提案された[2]。「突然変異」も形質遺伝を重視したものが出来ないかと考え、新しい『交換型突然変異』を考案し実験を試みた。対象として実在の合成樹脂工場の切り替えコスト最小問題を例にとり、改めて新しい遺伝子操作「交換型突然変異」を行い、順序表現による従来の突然変異で試行した結果と比較した。

結果は一部、〈JCKBSE 2000〉において発表したが[1]、まだその時点で有意差のある結果は出ていなかった。（以下、この研究を文献1という。）今回系統立てた実験をすることにより、『交換型突然変異』が有効なケースを明確にした。

2。遺伝的アルゴリズムとは

遺伝的アルゴリズムとは、色々な計画立案に対する生成検査法の一つである。どのように計画を作成すればいいか、その手順（アルゴリズム）がよく分からないが、でき上がった計画を評価する手段は持っているという事例に適用するとうまくいく時がある。

やりかたは、最初に何らかの方法で多数の案を作って見る。この研究の場合は、乱数を発生させることによって作った。

一つ一つの案を『個体』と言う。そして、案の中味一つ一つ（「どの様な順序で計画を遂行するか」における計画順序など）を『遺伝子』と言う。遺伝子はその並びが重要な意味を持つので、特に何番目の遺伝子かを問題にするとき、そこを『遺伝子座』という。実行不可能な計画（個体）は、発生の過程で存在を消す。これを『致死遺伝子』と呼ぶ。

次に、各個体が目的に添ったものかどうかを評価する。評価結果を『適応度』と言う。適応度の高いものは次の計画の時、なるべく多く発生するように、そうでないものはあまり発生させないように細工をして、次の『世代』を作成する。これを『選択』と言う。

このままでは、新世代は前の世代のコピーにすぎないのでここで二つの操作をする。

一つは、二つの個体が持っている遺伝子列を途中で入れ替えて見るのである。そうすると、前の世代にはなかった新しい個体が発生する。この中には、前の世代より適応度の高いものが発生する可能性がある。この操作を『交叉』と言う。

あと一つは、個体の中の、一部の遺伝子を全く違うものに入れ替えて見る。以前にはなかった個体が出現することは交叉と同じである。この操作を『突然変異』と呼ぶ。

新しい個体群（これを『世代』と言う。）についても、選択・交叉・突然変異を繰り返す。幾世代か後に、満足のいく案（個体）が発生することがある。

このような操作がまるで生物界の出来事みたいなので、遺伝的アルゴリズムと名付けられた。

3。スケジューリング問題の概要

3.1　本研究におけるスケジューリング問題の一般的定義

我々は、ここで扱うスケジューリング問題を以下のような性質を持った最適化問題とする。

ある一定の期間内に、決められた複数の製品をただ一度だけ製造する。決められた製品は必ず製造しなければならない。

問題の解：製造すべき製品をいかなる順序で割当てるか。
目的関数：切り替えコストを最小にすること。
制約条件：製品同士の製造順序の違いによる切り替えコストの発生。

3.2　本研究でのスケジューリング問題の性質

ここで扱うスケジューリング問題では、期間内に一度、必ず作らなければならない。また一度作った製品はその期間内にはそれ以上作らない。また逆に、作らなければならないものは既に決まっている。つまり利益最大型の問題ではなく、コスト最小型の問題である。ここで、一般の適応度比例戦略をとるには不都合がある。正解の値が分からないため正解とコストとの比を適応度としては使えないからである。また、適応度を上げると考えるよりも、コストの絶対値を減らすと言ったほうが考えやすい。

そこで、以前の研究では適応度をそのままは使わず、コストをペナルティとして捉え、ペ

ナルティの絶対値を評価するようにした[3][4]。

コストで計れない損失や、実行不可能な案も当然考えられるが、これも設計上の工夫でカバーする。

以上の性質を考えて以下の設計を行った。

3.3 多品種少量生産工場の製造スケジューリング

対象とする製品は、ある合成樹脂の生産計画である。品種は18銘柄からなる。製品の製造順序により切り替えコストが決まっている。

製造する銘柄に一連番号をつけて、製造順序とする。

先ず遺伝子表現だが、1個体を一つのスケジュール案とする。割当ての対象とするものの名前（製造する製品名）を遺伝子とし、遺伝子の並びは製造の順序を表わす。遺伝子長は製造対象製品の数になる。

次に適応度については、ペナルティを求めたあと、その世代の中の、最大のペナルティ値を持った個体から各個体のペナルティを引き、それを各個体の適応度とする。

こうすると、ペナルティ最大の個体については、適応度は0となる。選択戦略はルーレット方式とするため、次世代の発生確率は0である。これを逆エリート戦略と名付ける。

また、これでは、同じ計画案でも世代により適応度が異なるので、案の評価には適応度ではなくペナルティを用いる。

次にコストで計れない大きな損失を伴う計画案には、大きなペナルティ恣意的に与える。例えばある製品をこの時点で製造したら、納期遅れを生じ顧客の信用を失うなどである。このような大きなペナルティを、ペナルティ格差と名付けた[3][4]。

3.4 文献１により得られた知見と今回の実験の考え方

文献1では、GAパラメータや、その他のこのシステム固有の条件については、以下の数値及び考え方を使った。

(a) 個体数　100　世代数　1000

(b) 初期個体の集団によって、結果が大きく違うことも考え、乱数の発生を変えて10通り実行する。

(c) 切り替え難易度（切り替えコスト）は、0から129までのグループと、反応槽の全面清掃

が必要な100,000のグループに分ける。先ず後者の切り替えを最小にするように計画する。

(d) 納期の関係で特別に月初めに製造しなければならない銘柄については、納期遅れのペナルティを、更に10倍、1,000,000とした。

(c)及び(d)について、ペナルティに格差をつけたのは、その様な切り替えは可能な限り避けるためである。しかし致死遺伝子として全部排除してしまうと、計画そのものが出来なくなる可能性がある。

(e) 18銘柄を製造する。早期に製造しなければならない銘柄は、4銘柄である。この4銘柄を最初の6銘柄までに作らなければペナルティ(d)を課した。

実際に稼働している現在のシステム(ES)での出力結果は、ペナルティが400038、だった。

以上で実験した結果、次のことが判明した。

(1) すべての個体に『交換型突然変異』を施すと、却って悪い結果になった。

これについては、以下のように推測した。

「交換型突然変異」を単独で適用した結果が、思うようにならなかったのは、突然変異の性格から次の様な理由が考えられる。

前の遺伝子の形質を残して突然変異をするということは、前の遺伝子の影響が強いということである。確かに前の遺伝子がかなり良い性質を持っていて、その性質を利用してよりよい結果を導く時には有効だが、余り適応性に優れない遺伝子の形質を残しても意味がない。そこで、両者の特徴を生かして適応度の高い遺伝子には「交換型突然変異」を、そうでない遺伝子には順序型突然変異を試みた。

これにより、従って以上の共通条件の下に、以下の2ケース(①順序表現による突然変異、②交換型突然変異と順序表現による突然変異の併用)を実行した。

(2) 1000世代を実行したら、両案の差がほとんどなかった。また、400世代で打ち切ると②案が良かった。

これについては、こう考えられる。1000世代も実行すると両案とも、最適解に到達してしまうので、差が出ない。400世代で打ち切ると解が得られるには至らないことが多かったので差が出た。そこで、今回は、世代を決めるのではなく、満足解に到達するまでの必要世代を求めて比較した。

また、有意差が求まるように統計量を増やし、乱数を100回変えて実行した。

以上の実行プログラムはC言語で書かれた。プログラムは

http://www.nuis.ac.jp/~hig/kiyou1.html に示す。

3.5 実行システムの説明

3.5.1 順序表現による突然変異。

突然変異率を0.5とする。これにより発生した個体の半分が変異をとげる。これは、順序表現（P-TYPE）による変異とする。遺伝子型（G-TYPE）による変異だと、変異した途端に致死遺伝子となるからである。どの部分が変異するかは乱数によった。以前の研究では交叉と対になって両方行ったが[3][4]、文献1および本研究では比較のため、突然変異のみの操作をした。

全試行で、適応度400000以下となるまで世代を繰り返した。これはESの実用システムより良い結果が得られるまでという意味である。これを実験1と呼ぶ。

また、最適解が200050以下であることが確認されているし、これ以下なら極めて理想的な解であるので、200050以下になるまで同じ実験を繰り返すことも試みた。これを実験2と呼ぶ。

3.5.2 『交換型突然変異』

重複選択が出来ないスケジューリング問題では、G-TYPEによる突然変異を行うと、同じ製品を二回製造することになり、致死遺伝子となってしまうので、一般には、3.5.1のように、順序表現による突然変異を行う。しかし、特に遺伝子長が長い時、もとの遺伝子並びがバラバラになってしまうのが気になる問題であった。

例えば10製品の生産順序を求める問題で、ＡＢＣ・・・Jを製品名として、“ＩＡＣＤＦＪＨＢＧＥ”という生産順序を表わす個体の3番目の遺伝子に突然変異が起こったとする。

順序表現では“９１２２３５４１２１”となり、これが“９１７２３５４１２１”になったとすれば、G-TYPEに戻すと、“ＩＡＨＣＥＪＧＢＦＤ”になり、変異前の遺伝子の、4番目以降の遺伝子並び、“ＤＦＪＨＢＧＥ”の形質は全く残ってない。

そこで、G-TYPEのまま突然変異をさせ、上の例のように、3番目の遺伝子が変わり、“ＩＡＨＤＦＪＨＢＧＥ”になったとする。このままでは致死遺伝子なので、変異しなかった7番目の遺伝子Hを、3番目の遺伝子の変異前のもの、Cに置き換える。つまり、3番目と7番目の遺伝子を交換する。

“ＩＡＨＤＦＪＣＢＧＥ”となり、生存可能になるばかりではなく、4番目以降の遺伝子並びの

うち、“DFJ”と“BGE”が変異前の形質を残している。

この方法の長所は、突然変異前の遺伝形質を残すだけでなく、サブツアー交叉のように交叉の条件が厳しくなく、必ず突然変異を遂げられるということである。

この遺伝子操作を「交換型突然変異」と名付け、文献1で初めて適用を試みた。

ただ、この操作の単独使用は却って悪い結果になったのため、次項のように、両者の併用を試みた。

3.5.3 両者の併用

「交換型突然変異」を単独で適用した結果が、思うようにならなかったのは、突然変異の性格から次の様な理由が考えられる。

前の遺伝子の形質を残して突然変異をするということは、前の遺伝子の影響が強いということである。確かに前の遺伝子がかなり良い性質を持っていて、その性質を利用してよりよい結果を導く時には有効だが、余り適応性に優れない遺伝子の形質を残しても意味がない。そこで、両者の特徴を生かして適応度の高い遺伝子には「交換型突然変異」を、そうでない遺伝子には順序型突然変異を試みた。

そこで、適応度がいくつ以下なら形質を残すべきか、2つの適応度について試みた。

実験1については、適応度が400000以下なら解としたので、適応度500000以下と600000以下の2ケースについて「交換型突然変異」を適用した。

実験2については、200050以下を解としたので、400000以下と500000以下の2ケースについて実験した。

試行は、3.5.1と同じ100回である。

3.5 システムの実行結果

実験1の実験結果を（表1）に示す。

これによると、解到達までの平均世代数は、ほとんど差が出なかった。しかし、100回の実験を、個別に比較してみると、順序表現と500000以下を比較すると、500000以下側から見て、44勝31敗25分になっている。この意味は、「交換型突然変異」を利用したほうが、速く解に到達したのが44回、かえって遅くなるのが31回、25回は突然変異のやり方に関係なく同じ結果だったと言える。

ここで25回も引き分けがあるということは、「交換型突然変異」が解の早期発見に全く影響を与えなかったということで、それはそれで意味のあることではある。

しかし、ここでは解の勝敗について考える。つまり、「交換型突然変異」が解の早期発見に寄与したかどうかである。44勝31敗は勝率5割8分7厘である。これを勝率5割と仮定して検定する。これは、近似的に正規分布 $N(37.5、4.33^2)$ に従うので、7%で仮説は棄却される。

また、これだけ有意差があるにもかかわらず、平均到達世代数があまり変わらないのは、ケース6、10、63で両者の到達世代差が100以上あって、その3ケースだけで平均を大きく引き上げているからである。これは、順調に行けは有効な「交換型突然変異」が、たまに迷路に入り込み、出口を探すのに苦労している様が見てとれる。

実験2の実験結果を（表2）に示す。

これによると、解到達までの平均世代数は、従来の順序表現による方がかえって良い結果となっている。これは筆者の提案する「交換型突然変異」が有効でなかったことを意味する。

4。結果の考察

前項で述べた通り、「交換型突然変異」のほうが、順序表現による突然変異より、全部良かったわけではない。

しかし全部の遺伝子が前の形質を残さない方がいいかというと、決してそうではなく、ある程度良い適応度の遺伝子については残した方が良い結果が出るケースがあることが判明した。

今回は単純に適応度だけでどちらの突然変異を選ぶかを決めた。しかし今回は試行しなかったが、例えば残したい遺伝子並びを部分的に持った個体（その部分の切り替えコストが極小に近いもの）については、適応度が悪くても形質を残すなどの工夫をすることで、更に効果的な遺伝子操作が出来るのではないかと考えられる。

また、遺伝子の並びの形質を残すということは、巡回セールスマン問題など他の応用分野にも適用されることである。特にJCKBSE 2000では、「ジョブショップスケジューリングに適用しては」という意見も出された。

二つの突然変異を使い分けることにより、良い結果がでたので、今後もそういう対象を探して、この交換型突然変異の効用をさらに求めていきたい。

参考文献

(1) M.Higuchi, M.Nagata:「Applying of Character-Preserving Mutation to Scheduling Problem」Knowledge-Based Software Engineering, pp59-64, 2000

(2) 山村雅幸、小野貴久、小林重信：「形質遺伝を重視した遺伝的アルゴリズムによる巡回セールスマン問題の解法」人工知能学会誌、Vol.7, No6, pp.117-127, 1992

(3) 樋口光明、永田守男：「多品種少量生産工場の製造スケジューリングに対するGAの適用」情報処理学会全国大会講演論文集 5K-3, 1995

(4) M.Higuchi, M.Nagata:「An Application of the Genetic Algorithm to Scheduling Problems Using the Concept of Differential Penalty」Second Joint Conference on Knowledge-Based Software Engineering, pp202-205, 1996

表1　400000以下を解到達とする結果〔実験1〕

実験回数	乱数の初期値	解到達までの世代数			500000以下での結果			600000以下での結果		
		順序表現	500000	600000	勝	分	負け	勝	分	負け
1	2	137	136	114	1	0	0	1	0	0
2	3	46	45	42	1	0	0	1	0	0
3	5	104	48	50	1	0	0	1	0	0
4	7	20	20	31	0	1	0	0	0	1
5	11	33	33	42	0	1	0	0	0	1
6	13	40	167	52	0	0	1	0	0	1
7	17	74	66	55	1	0	0	1	0	0
8	19	37	21	17	1	0	0	1	0	0
9	23	110	123	117	0	0	1	0	0	1
10	29	41	180	248	0	0	1	0	0	1
11	31	39	39	57	0	1	0	0	0	1
12	37	22	29	39	0	0	1	0	0	1
13	41	71	84	97	0	0	1	0	0	1
14	43	14	30	23	0	0	1	0	0	1
15	47	71	60	12	1	0	0	1	0	0
16	53	16	16	56	0	1	0	0	0	1
17	59	46	40	17	1	0	0	1	0	0
18	61	43	59	54	0	0	1	0	0	1
19	67	26	26	22	0	1	0	1	0	0
20	71	24	24	24	0	1	0	0	1	0
21	73	27	24	38	1	0	0	0	0	1
22	79	47	64	250	0	0	1	0	0	1
23	83	17	18	18	0	0	1	0	0	1
24	89	51	49	62	1	0	0	0	0	1
25	97	38	37	76	1	0	0	0	0	1
26	101	23	23	20	0	1	0	1	0	0
27	103	37	88	115	0	0	1	0	0	1
28	107	33	34	76	0	0	1	0	0	1
29	109	107	132	240	0	0	1	0	0	1
30	113	25	25	56	0	1	0	0	0	1
31	127	41	41	71	0	1	0	0	0	1
32	131	60	32	50	1	0	0	1	0	0
33	137	34	34	34	0	1	0	0	1	0
34	139	89	85	79	1	0	0	1	0	0
35	149	61	9	9	1	0	0	1	0	0
36	151	45	35	74	1	0	0	0	0	1
37	157	51	38	30	1	0	0	1	0	0
38	163	50	50	87	0	1	0	0	0	1
39	167	19	19	19	0	1	0	0	1	0
40	173	120	135	61	0	0	1	1	0	0
41	179	138	130	60	1	0	0	1	0	0
42	181	21	21	21	0	1	0	0	1	0
43	191	123	112	103	1	0	0	1	0	0
44	193	68	43	30	1	0	0	1	0	0
45	197	20	20	15	0	1	0	1	0	0
46	199	127	104	30	1	0	0	1	0	0
47	211	56	56	65	0	1	0	0	0	1
48	223	15	13	13	1	0	0	1	0	0
49	227	51	44	39	1	0	0	1	0	0
50	229	75	78	159	0	0	1	0	0	1
51	233	33	32	53	1	0	0	0	0	1

実験回数	乱数の初期値	解到達までの世代数		
		順序表現	500000	600000
52	239	85	83	50
53	241	104	41	58
54	251	19	61	27
55	257	71	62	62
56	263	37	43	51
57	269	54	71	80
58	271	40	84	29
59	277	44	47	39
60	281	30	91	40
61	283	49	57	66
62	293	106	50	40
63	307	23	169	62
64	311	113	42	37
65	313	43	43	56
66	317	33	16	22
67	331	75	43	67
68	337	123	65	89
69	347	40	40	207
70	349	27	27	27
71	353	53	76	69
72	359	75	48	54
73	367	58	58	34
74	373	40	40	27
75	379	45	45	84
76	383	61	46	78
77	389	40	40	48
78	397	70	43	37
79	401	27	25	32
80	409	23	23	20
81	419	46	105	48
82	421	80	95	31
83	431	56	49	111
84	433	27	27	38
85	439	32	35	51
86	443	71	22	40
87	449	36	36	57
88	457	61	123	54
89	461	101	64	100
90	463	121	51	29
91	467	102	118	45
92	479	84	65	78
93	487	76	56	35
94	491	39	80	60
95	499	75	109	50
96	503	46	29	29
97	509	67	73	72
98	521	47	36	40
99	523	55	42	62
100	541	54	18	18
	平均	56.0	56.8	58.6

500000以下での結果			600000以下での結果		
勝	分	負け	勝	分	負け
1	0	0	1	0	0
1	0	0	1	0	0
0	0	1	0	0	1
1	0	0	1	0	0
0	0	1	0	0	1
0	0	1	0	0	1
0	0	1	1	0	0
0	0	1	1	0	0
0	0	1	0	0	1
0	0	1	0	0	1
1	0	0	1	0	0
0	0	1	0	0	1
1	0	0	1	0	0
0	1	0	0	0	1
1	0	0	1	0	0
1	0	0	1	0	0
0	1	0	0	0	1
0	1	0	0	1	0
0	0	1	0	0	1
1	0	0	1	0	0
0	1	0	1	0	0
0	1	0	1	0	0
0	1	0	0	0	1
1	0	0	0	0	1
0	1	0	0	0	1
1	0	0	1	0	0
1	0	0	0	0	1
0	1	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	0
1	0	0	0	0	1
0	1	0	0	0	1
0	0	1	0	0	1
1	0	0	1	0	0
0	1	0	0	0	1
0	0	1	1	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	0	1	1	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	0
1	0	0	1	0	0
0	0	1	0	0	1
1	0	0	1	0	0
1	0	0	0	0	1
1	0	0	1	0	0
44	25	31	48	5	47

表2　2000050以下を解到達とする結果〔実験2〕

実験回数	乱数の初期値	解到達までの世代数			400000以下での結果			500000以下での結果		
		順序表現	400000	500000	勝	分	負け	勝	分	負け
1	2	302	176	147	1	0	0	1	0	0
2	3	416	100	52	1	0	0	1	0	0
3	5	141	142	305	0	0	1	0	0	1
4	7	402	262	97	1	0	0	1	0	0
5	11	82	92	150	0	0	1	0	0	1
6	13	274	279	496	0	0	1	0	0	1
7	17	275	659	170	0	0	1	1	0	0
8	19	101	208	266	0	0	1	0	0	1
9	23	289	452	245	0	0	1	1	0	0
10	29	197	295	659	0	0	1	0	0	1
11	31	425	65	70	1	0	0	1	0	0
12	37	281	54	126	1	0	0	1	0	0
13	41	323	204	140	1	0	0	1	0	0
14	43	417	126	487	1	0	0	0	0	1
15	47	296	568	612	0	0	1	0	0	1
16	53	57	246	54	0	0	1	1	0	0
17	59	56	63	55	0	0	1	1	0	0
18	61	46	46	417	0	1	0	0	0	1
19	67	220	75	112	1	0	0	1	0	0
20	71	231	320	691	0	0	1	0	0	1
21	73	135	55	713	1	0	0	0	0	1
22	79	249	324	188	0	0	1	1	0	0
23	83	255	230	272	1	0	0	0	0	1
24	89	149	305	78	0	0	1	1	0	0
25	97	72	55	64	1	0	0	1	0	0
26	101	82	271	55	0	0	1	1	0	0
27	103	396	340	199	1	0	0	1	0	0
28	107	133	93	154	1	0	0	0	0	1
29	109	284	429	560	0	0	1	0	0	1
30	113	68	1908	175	0	0	1	0	0	1
31	127	250	590	173	0	0	1	1	0	0
32	131	422	135	70	1	0	0	1	0	0
33	137	379	74	69	1	0	0	1	0	0
34	139	270	417	632	0	0	1	0	0	1
35	149	171	154	30	1	0	0	1	0	0
36	151	58	61	71	0	0	1	0	0	1
37	157	156	988	554	0	0	1	0	0	1
38	163	67	525	403	0	0	1	0	0	1
39	167	260	430	34	0	0	1	1	0	0
40	173	168	168	372	0	1	0	0	0	1
41	179	242	232	234	1	0	0	1	0	0
42	181	431	294	163	1	0	0	1	0	0
43	191	576	446	1592	1	0	0	0	0	1
44	193	190	117	117	1	0	0	1	0	0
45	197	55	58	39	0	0	1	1	0	0
46	199	174	173	112	1	0	0	1	0	0
47	211	174	232	80	0	0	1	1	0	0
48	223	132	226	50	0	0	1	1	0	0
49	227	112	176	707	0	0	1	0	0	1
50	229	355	1263	84	0	0	1	1	0	0
51	233	57	85	132	0	0	1	0	0	1

実験	乱数の	解到達までの世代数			400000以下での結果			500000以下での結果		
回数	初期値	順序表現	400000	500000	勝	分	負け	勝	分	負け
52	239	169	198	190	0	0	1	0	0	1
53	241	494	202	140	1	0	0	1	0	0
54	251	19	19	190	0	1	0	0	0	1
55	257	146	169	84	0	0	1	1	0	0
56	263	755	561	820	1	0	0	0	0	1
57	269	268	311	183	0	0	1	1	0	0
58	271	154	238	279	0	0	1	0	0	1
59	277	101	77	67	1	0	0	1	0	0
60	281	267	117	205	1	0	0	1	0	0
61	283	615	131	74	1	0	0	1	0	0
62	293	331	272	119	1	0	0	1	0	0
63	307	56	299	216	0	0	1	0	0	1
64	311	141	189	632	0	0	1	0	0	1
65	313	67	87	107	0	0	1	0	0	1
66	317	146	233	1204	0	0	1	0	0	1
67	331	678	154	211	1	0	0	1	0	0
68	337	263	405	376	0	0	1	0	0	1
69	347	352	809	186	0	0	1	1	0	0
70	349	83	42	62	1	0	0	1	0	0
71	353	241	188	175	1	0	0	1	0	0
72	359	105	191	147	0	0	1	0	0	1
73	367	245	626	400	0	0	1	0	0	1
74	373	119	751	149	0	0	1	0	0	1
75	379	298	633	164	0	0	1	1	0	0
76	383	157	275	60	0	0	1	1	0	0
77	389	124	297	194	0	0	1	0	0	1
78	397	170	90	330	1	0	0	0	0	1
79	401	53	53	69	0	1	0	0	0	1
80	409	86	393	269	0	0	1	0	0	1
81	419	127	149	520	0	0	1	0	0	1
82	421	975	186	128	1	0	0	1	0	0
83	431	341	149	731	1	0	0	0	0	1
84	433	48	107	883	0	0	1	0	0	1
85	439	56	87	144	0	0	1	0	0	1
86	443	151	334	395	0	0	1	0	0	1
87	449	63	136	161	0	0	1	0	0	1
88	457	261	539	205	0	0	1	1	0	0
89	461	178	421	705	0	0	1	0	0	1
90	463	284	190	160	1	0	0	1	0	0
91	467	357	389	649	0	0	1	0	0	1
92	479	244	374	673	0	0	1	0	0	1
93	487	170	104	250	1	0	0	0	0	1
94	491	435	391	179	1	0	0	1	0	0
95	499	165	317	149	0	0	1	1	0	0
96	503	70	114	104	0	0	1	0	0	1
97	509	105	105	297	0	1	0	0	0	1
98	521	95	1177	217	0	0	1	0	0	1
99	523	76	109	108	0	0	1	0	0	1
100	541	37	54	37	0	0	1	0	1	0
	平均	222.9	287.3	275.2	36	5	59	47	1	52