卒業論文

完全グラフ表現による遺伝的アルゴリズム

平成 6 年 2 月 16 日 提出

指導教官　　田中 英彦 教授
　　小池 帆平 講師

東京大学 工学部 電子工学科
20516　　井手 一郎

内容梗概

従来の遺伝的アルゴリズムでは基本的に、交叉は一次元表現された染色体をランダムに切断することにより行なわれていた。一般に、染色体上の各遺伝子座には、生存していく上で重要な遺伝子を保有するものと、そうでないものとがある。重要な遺伝子を保有する遺伝子座が複数あるとき、それらを交叉の際に散逸させるのは、進化を進める上で非効率的である。そこで、染色体を完全グラフによって表現し、そこに各遺伝子座の重要度に応じて決定された全遺伝子座間の結合係数を記述した。そして、重要な遺伝子を保有する遺伝子座間での切断が起きにくくなるように、結合係数の値に応じた確率で切断を行うようにした。このような工夫の結果、適用する問題によっては、収束値や収束速度に関して効果が認められた。

目 次

図 目 次

第 1 章

序論

1.1　研究の目的

従来の遺伝的アルゴリズムでは、染色体は遺伝子が一次元に並んでいるものとして扱われ、それゆえに、染色体上に重要な遺伝子が散らばっている場合、交叉によって簡単に散逸するという非効率的な手法が取られてきた。そこで、本研究ではこのような散逸を防ぐために、染色体を完全グラフとして扱い、各ノード間の結合係数を用いて、重要な遺伝子同士を固めるようにした。このような手法が実現できるならば、交叉の際の切断は重要な遺伝子間では起こりにくくなり、効率的な進化が図れるだろう。

以上のような考えのもとに、完全グラフ表現を用いて遺伝的アルゴリズムを改良すること[1]、そしてどのような問題に対して特に有効かを調べるのが、本研究の目的である。

1.2　本論文の構成

まず第 2 章で遺伝的アルゴリズムについて簡単に記し、続いて第 3 章で染色体を完全グラフ表現することによる遺伝的アルゴリズムの改良について記す。そして、第 4 章で実際に行った種々のシミュレーションと、その結果について記した後、第 5 章で結論と考察をまとめる。

[1] 以下、染色体の完全グラフ表現を用いた交叉を、**完全グラフ交叉**と呼ぶ。

第 2 章

遺伝的アルゴリズム

2.1　遺伝的アルゴリズムの原理

今日の生物学では、生物は長年の自然淘汰や突然変異を経て、現在の姿に進化したと考えられている。自然界では、淘汰は生物を取り巻く環境の変動によって起きる。環境に適応できた個体 (あるいは種) は生き残り、そうでないものは滅びてゆく。ここで注目したいのは、生存できた個体の遺伝子は、意図的に環境に適応できるように作られたのではなく、交配や突然変異により生存に有利な性質を偶然もつようになり、その結果、個体の生存が許されたということである。

このような自然界のアルゴリズムを真似て、ある問題の最適解を与えるコードの効率的な探索に応用できないか、という発想をもとにして**遺伝的アルゴリズム**が考えられた。自然界では生物を取り巻く環境によって淘汰が起きるが、人工的な (計算機上の) 環境では、問題に応じた評価関数を用いて淘汰 (選択) を行う。

2.2　用語の定義

遺伝的アルゴリズムについて記すにあたって、予め用語の定義をしておく。

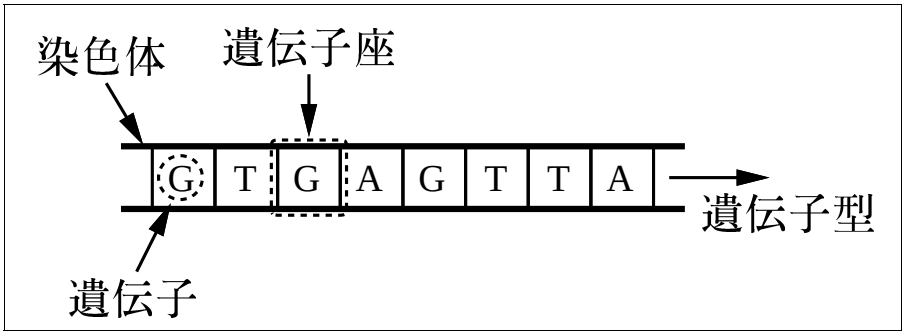


図 2.1: 用語の定義

- **染色体** (*chromosome*)

  遺伝情報を伝える実態。

  生体では、塩基 (*base*) で構成された物理的実態であるが、遺伝的アルゴリズムでは、各個体の遺伝情報を蓄える配列である。

- **遺伝子座** (*locus*)

  染色体上の位置 (座標)。

  通常は、各遺伝子座に記述される遺伝情報の性質は決まっている。

  配列でいえば、添字に相当する。

- **遺伝子** (*gene*)

  各遺伝子座に蓄えられた遺伝情報。

  各遺伝子座に記述される遺伝情報の性質は決まっているので、そこに記述される遺伝情報 (=遺伝子) によって、個体の種々の遺伝的特徴が決定される。生体では T(チミン), A(アデニン), G(グアニン), C(シトシン) の 4 種の塩基が遺伝子になる。

  配列でいえば、各要素に格納されているデータ (数値、文字列など) に相当する。

- **遺伝子型** (*genotype*)

  ある染色体上の遺伝子の列。

2.3 全体の処理の流れ

実際のアルゴリズムの基本的な流れは、図 2.2に示すとおりである。

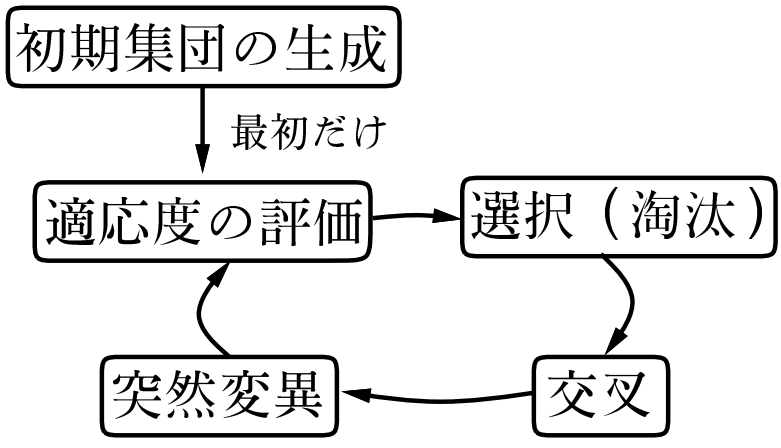


図 2.2: 遺伝的アルゴリズムの処理手順

以下、それぞれの処理について記す。

2.4 各部分の処理

2.4.1 初期集団の生成

初期集団の遺伝子は乱数によって生成する。平均的な動作を見るためには、異なった乱数系列によって生成される異なった初期集団から始める実験を複数回行うことになる。

第 4 章で行うすべての実験の結果は、10 通りの乱数系列による実験の平均値である。

2.4.2 適応度の評価

この処理は、解こうとする問題に応じて異なる。ある問題において、よい性質を示す遺伝子型をもつ個体には高い評価値を与える。

評価値と重要度の関係を非線型にすることによって、よい性質をもつ染色体の評価値を際立たせるなどの操作を行うこともできるが、本研究では評価関数の値をそのまま用いた。

2.4.3 選択 (淘汰)

前の処理で評価された各個体はこの処理で選択される。ある決められた閾値に対して、それ以上の評価値をもつ染色体は次世代での生存を許され、それ未満のものは淘汰されるのである。閾値の決め方には、絶対的な値を予め決めておく方法のほかに、その世代の最良値や平均値の何 % かに決め、毎世代異なった値にする相対的な方法がある。真の最適解があらかじめ分かっているときには前者を用いることができるが、そうでないときは後者を用いるとよい。

2.4.4 交叉

前の処理で生き残った個体群の中から、評価値に応じた確率で選ばれた染色体の交配を行う処理である。この処理によって次世代の個体群が形成される。交叉前の「親」を残す手法と、残さない手法があるが、ここでは後者を採用した。

交叉は、図 2.3, 2.4に示すように、単純交叉と複数点交叉の 2 種類の手法に分類できる。

本研究の主題である完全グラフ交叉は、複数点交叉の一種である。

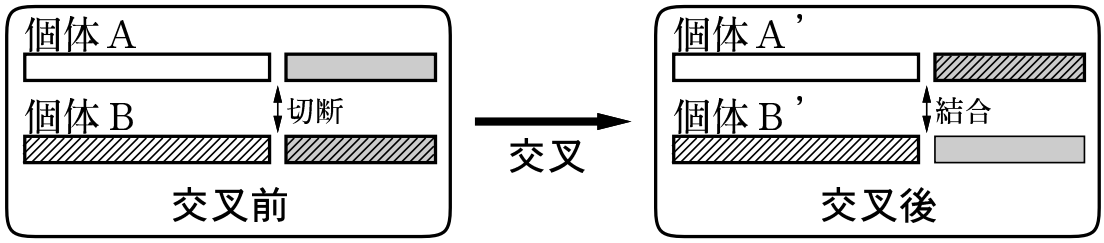


図 2.3: 単純交叉

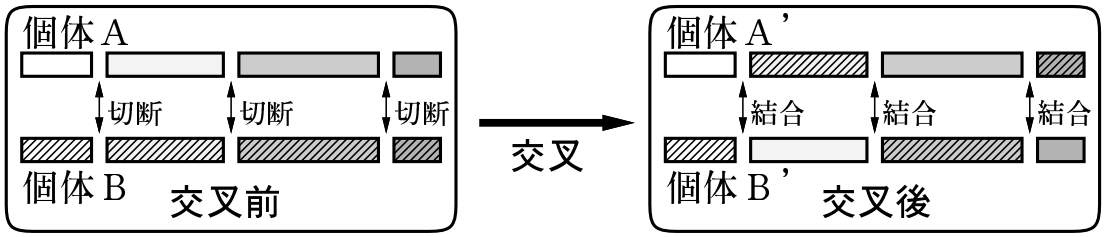


図 2.4: 複数点交叉 (図は 3 点交叉の場合)

2.4.5 突然変異

各遺伝子座毎に、ある確率でビットの反転や交換を起こす処理である。すでによい評価値をもっている遺伝子型を破壊する危険性がある反面、局所解に陥ったときに、そこから脱出することを可能にする利点がある。

第 3 章

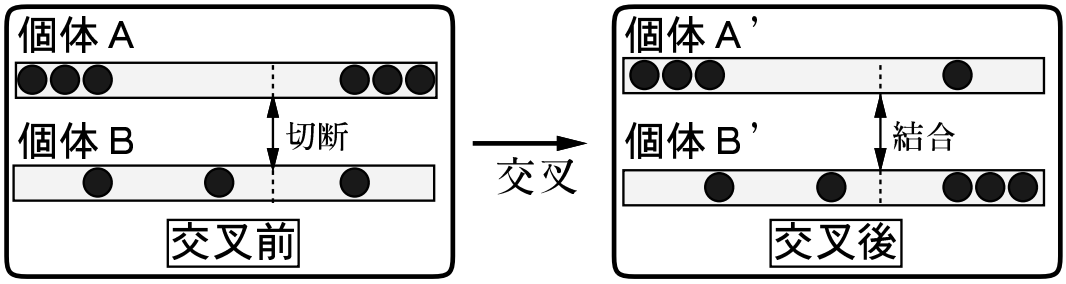
遺伝的アルゴリズムの改良

3.1　染色体の完全グラフ表現

3.1.1　従来の遺伝的アルゴリズムの問題点

2.4.4では、交叉の際の切断回数の違いによる遺伝的アルゴリズムの分類を示したが、ここでは切断場所の決定法について考えてみる。

従来の遺伝的アルゴリズムでは、染色体は一次元の遺伝子座の列で、それをランダムな位置で切断し、交叉していた。このようなランダムな交叉では、特定の (複数の) 遺伝子座に、評価値をよくする情報が存在していても、交叉の際に散逸してしまい、次世代には凡庸な個体が生じるにとどまってしまうことが多発すると考えられる (図 3.1)。このような現象は、進化の速度を遅くする要因の一つになっていると思われる。



●は、重要な遺伝子を表す。
このように、1 次元的な交叉では、交叉後も評価値はあまり変わらない。

図 3.1: 一次元的な交叉

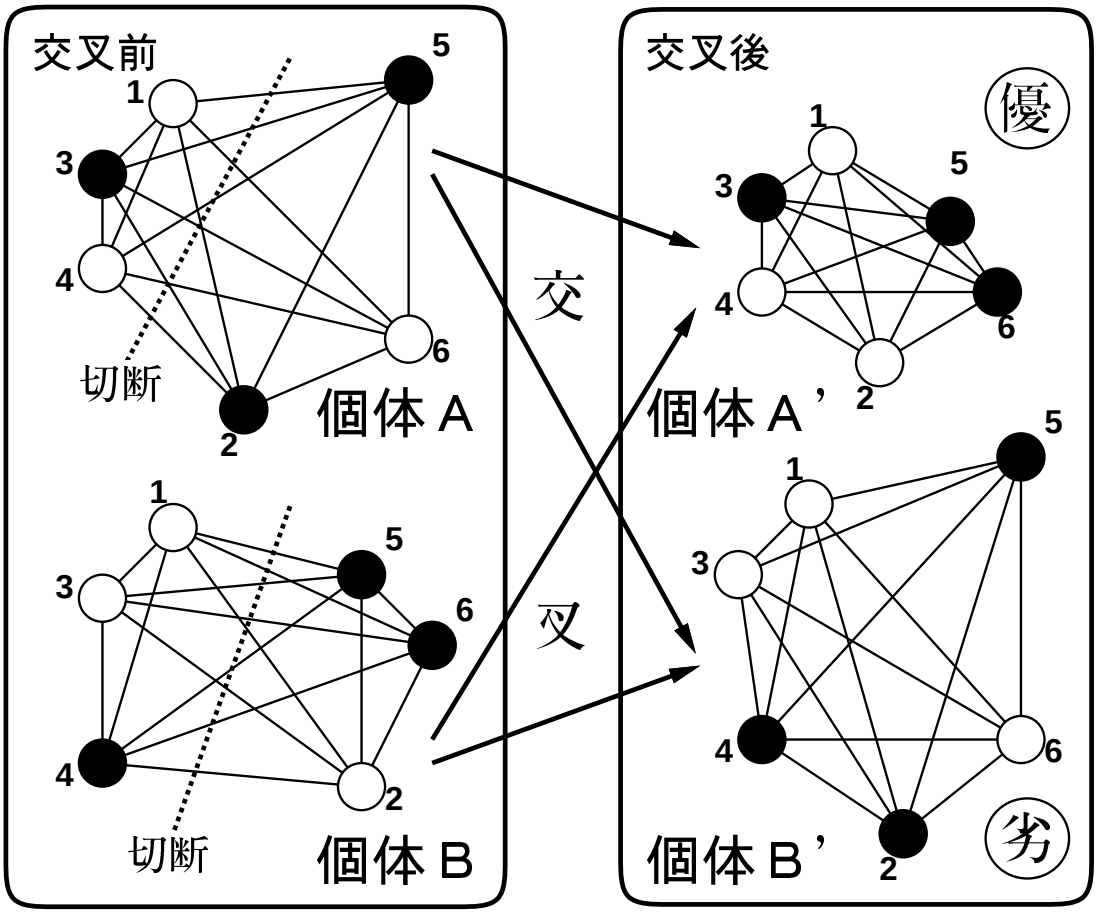
3.1.2　問題点の解決法

3.1.1で指摘した問題点を解決するには、まず、どの遺伝子座に評価値をよくする遺伝子が乗っているかを知ることが必要である。本研究では、よい評価値を得ている個体の各遺伝

子は、総じてよいものであると考えることにした。詳しいアルゴリズムは 3.2.2で述べる。

次に、評価値をよくする遺伝子が乗っている遺伝子座のグループと、そうでないもののグループとに分けて、その境界で交叉をすればよい。このようなグループ分けを行うためには、一次元的に隣接する遺伝子座間の組のみではなく、あらゆる遺伝子座間の結合係数を考えなければならない。このようなデータ表現は、完全グラフを用いることによって実現される (図 3.2)。

以上の考えをもとにしたシステムを構築することにより、交叉を行っても、評価値をよくする遺伝子が散逸しにくくなり、効率的な進化が図れるだろう。



注 1 ：図中の番号は遺伝子座の番号を示す

注 2 ：各遺伝子座（ノード）間の距離は結合係数の大きさに反比例する。
直観的に言えば、重要な遺伝子同士が集まって塊を作る。

図 3.2: 完全グラフを用いた交叉 (概念図)

3.2 完全グラフ表現による遺伝的アルゴリズム

3.2.1 全体の処理の流れ

基本的には図 2.2で示した処理手順と同じだが、図 3.3に示すように、完全グラフ表現をすることに伴って、「結合係数計算」の処理が加わる。また、完全グラフ交叉を行うことによって、交叉の処理が複雑になる。

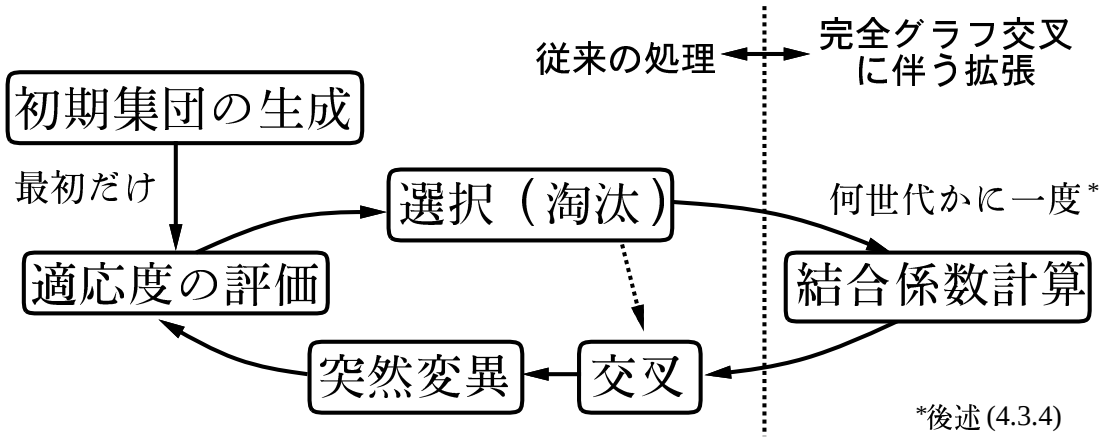


図 3.3: 完全グラフ交叉を用いた遺伝的アルゴリズムの処理の流れ

3.2.2 結合係数の計算

すでに記したが、交叉の際に、評価値をよくする遺伝子が乗っている遺伝子座間を切断したくない。ここで問題になるのは、評価値をよくする遺伝子をどのように知るかである。

ここでは、前世代の個体群を見て、図 3.4に示すようなアルゴリズムで結合係数を計算している。ある 1 組の遺伝子座間の結合係数は、それぞれの遺伝子座の内容 (0 か 1 か[1]) によって異なる。なぜならば、遺伝子座 a_i と a_j ($i \neq j$) 上の遺伝子の組が (0,1) なら重要だが、(1,1) なら重要でないということがあるからだ。そのために、結合係数を記述する表 (配列) を、遺伝子の可能な組み合わせの分 ((0,0),(0,1),(1,0),(1,1) のそれぞれの場合の 4 通り分) 用意しなければならない。

このように多くの配列を用意しなくてはならないことは、記憶容量の制限から、 3.3に記すような問題を生じるが、完全グラフ交叉を行うためには不可欠な処理である。

[1] 遺伝子が 2 値の場合。以下も簡便のため同様。

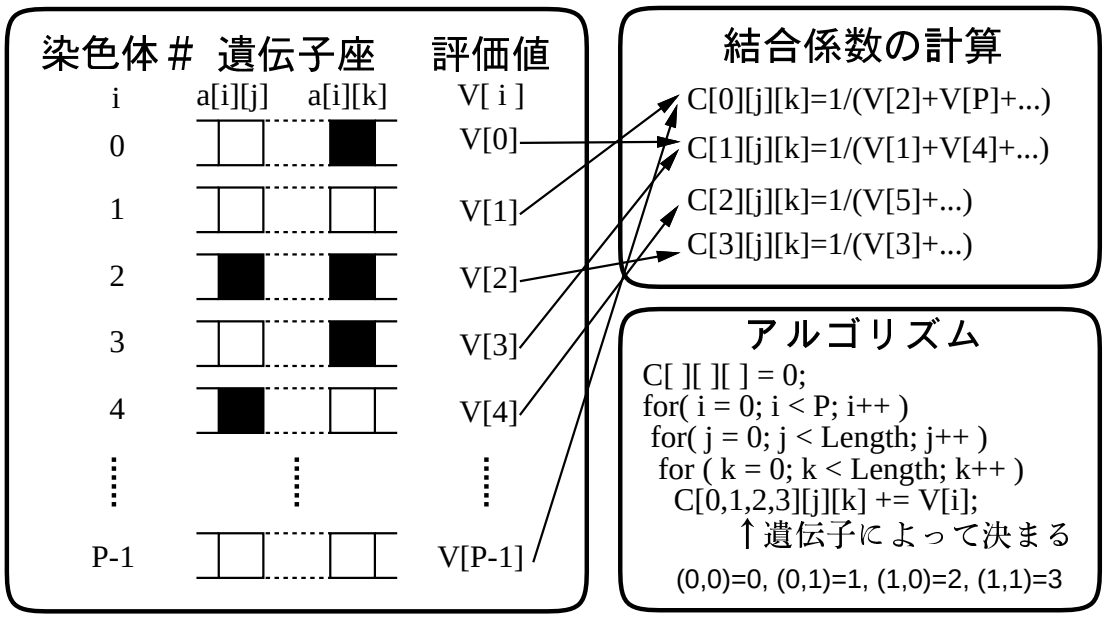


図 3.4: 結合係数計算のアルゴリズム

3.2.3 交叉の際の切断箇所の決定

切断を行う際には、その染色体の遺伝子座 a_i と a_j ($i \neq j$) 上の遺伝子に対応した表を参照する。ここで生じる疑問は、交叉を行おうとする 2 個の染色体では、遺伝子座 a_i と a_j ($i \neq j$) 上の遺伝子は通常異なるので、切断する場所の決定がうまくいかないのではないかということだ。それが図 3.4に示すアルゴリズムの問題点なのだが、実際には、交叉を行おうとしている両染色体の各遺伝子座間の結合係数の平均をとって、それを切断の際に参照している。

以上の準備段階を経て、実際の切断が行われる。実際の切断は、予め切断場所を決めて切断するというよりは、重要な遺伝子座 (完全グラフのノード) を 1 つづつ切り取っていくという手法を用いている。各ノードを切断するか否かは、各ノードから出る全ての枝の結合係数の和を計算し、その和の値の大きさに応じた確率によって決まる。完全グラフ交叉は簡単に言えば、よい性質をもつ遺伝子を集めることなので、結合係数の和の大きなものを重要な遺伝子だと考えると、それらを次々に集めていけばよいことになる。

3.3 本アルゴリズムの問題点

遺伝子の取りうる値が 2 値以上の場合にも、基本的には 3.2に記した原理に従えばよいのだが、結合係数を記録するために用意しなければならない配列の数が莫大になる (n 値なら、n^2個必要) ので、あまり多値の問題に適用することは、記憶容量の制限から現実的ではない。

さらに、連続値をとるような場合には、実数を近似的に離散値として扱えばよいが、純粋に解こうとすると、∞ 個の配列を用意しなくてはならず、実現不可能である。近似度はやはり計算機の記憶容量に依存する。

また、アルゴリズムの性質上、交叉やそのための前処理に時間がかかるのは自明で、結局は単純な遺伝的アルゴリズムの方が速く収束することも考えられる。

これらの点が本アルゴリズムの問題点である。しかし、用意できる配列の数は計算機の記憶容量にのみ依存するので、今後搭載記憶容量が増えるにつれ、大がかりな計算も可能になろう。

なお、後に行う n-Queens 問題の実験では、100 個体, n=32 でおよそ 8MBytes のメモリを消費していた。メモリの大部分を結合係数格納に費しているとすると、メモリ消費量は n の累乗に比例するので、64MBytes 搭載の計算機を用いた場合で n=90 程度が最大だと思われる。

処理時間の問題については、後程 4.3.4で議論する。

第 4 章

実験と評価

3.2で述べたような手法を実現するシステムを構築し、性能を評価するためにいくつかの実験を行った。また、単純交叉と複数点交叉による実験も比較のために並行して行った。

4.1 簡単な例題

4.1.1 例題について

次の 2 つの簡単な例題について実験を行った。

- **例題** 1
  評価関数は、遺伝子列中の 1 の個数の割合。$f = \sum_{i=1}^{L} \frac{a_i}{L}$ (L は遺伝子長,a_iは各遺伝子)
- **例題** 2
  パターンマッチング。評価関数は、目的とする遺伝子列と一致している遺伝子の割合。

評価関数の値が、例題 1 では遺伝子座に全く依存しない (単に 1 が多ければ多いほどよい) のに対して、例題 2 では極度に依存している点で、両者の性質は大きく異なる。
実験は、次のような条件のもとに行った。

表 4.1: 各種定数の設定値

定数	値
遺伝子	(0,1)
遺伝子長 (L)	30
個体数	100
世代当たりの交叉回数	50
遺伝子座当たりの突然変異率	0.1%
選択の際の評価値の基準	0.4

4.1.2 実験の結果と考察

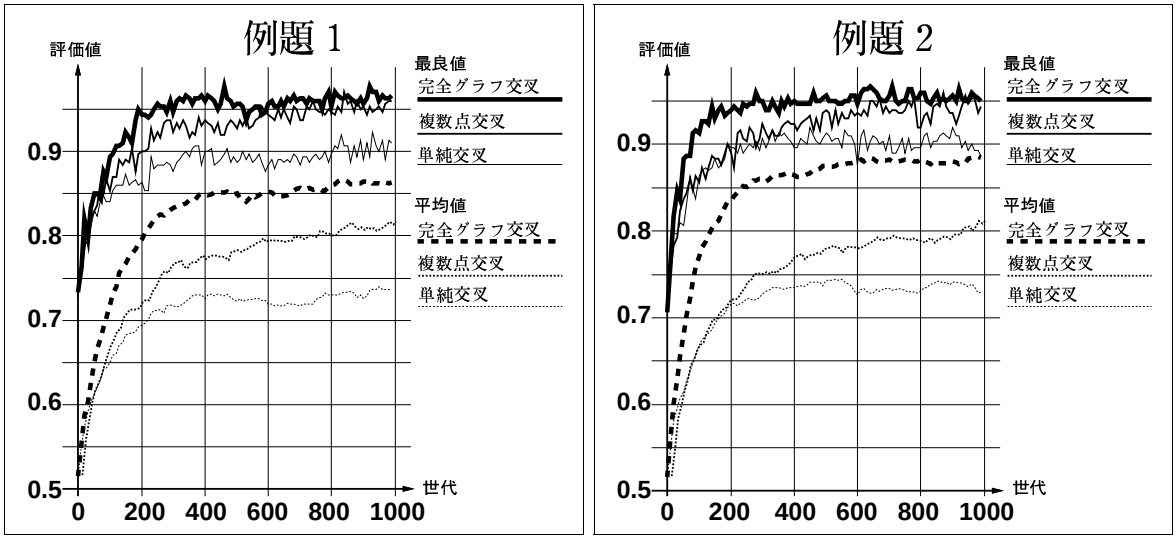


図 4.1: 例題 1, 2 の最良値と平均値の推移

結果を眺めるときに注目すべき点は、次に示す 3 点ほどがある。これらは、以下の他の例題についても同様である。

- グラフの立上り
- 収束値
- 収束速度

以上の点について順に眺めていくと、グラフの立上りは、例題 2 の方が例題 1 と比較して、かなり急峻になっている。収束値は完全グラフ交叉、複数点交叉ともに、両例題で同じような値に収束しているが、単純交叉と比較するとよい値に収束している。収束速度は、例題 2 では、完全グラフ交叉が他の手法よりかなり早いが、例題 1 では複数点交叉と大差ない速度である。

これらの結果は、基本的に前項の最後で述べたような、例題の性質に由来すると思われる。つまり、例題 1 は評価関数が遺伝子座に依存しないものであるために、完全グラフ交叉の威力が発揮されていない。一方、例題 2 は評価関数が遺伝子座に依存するものであるために、完全グラフ交叉の威力が発揮されているのである。

また、複数点交叉が最終的には完全グラフ交叉と同等の値に収束しているのは、完全グラフ交叉が複数点交叉の一種であることから説明できる。

以上のような結果になったが、本来は両例題ともに最適解である 1 に収束して欲しいわけである。ここで収束値が、0.95 という値にとどまっているのは、局所解に陥っているためというよりは、各世代での最良値をもつ染色体が、次世代に受け継がれる際に、突然変

異や交叉を受けてしまうことによると考えられる。そこで、そのような劣化を防ぐために、エリート保存戦略という手法が考え出されている。

エリート保存戦略とは、評価値が最良の染色体 (「エリート」と呼ぶ) については突然変異や交叉の操作を行わずに、次世代まで温存する手法である。

例題 1, 2 に適用した結果を図 4.2に示す。

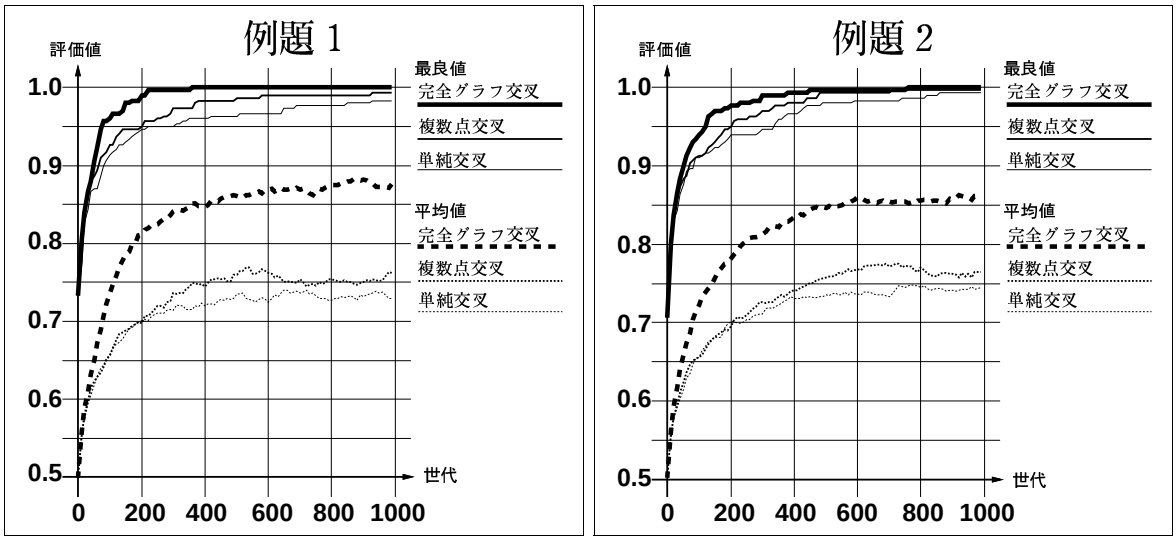


図 4.2: 例題 1, 2 の最良値と平均値の推移 (エリート保存)

これからも明らかなように、この工夫により、収束速度が高速化されたのとともに、完全グラフ交叉を用いた場合には、収束値が最適解である 1 に到達するようになった。

以上のようにエリート保存戦略を行うことによって、進化を効率化できることが分かったので、以降の実験ではこの戦略を採用することにした。

この手法を採用することによって、局所解に陥りやすくなる心配はほとんどない。評価値が最良の個体は自らは交叉には加わらず、子孫をつくらないし、自らの評価値を越える個体が出現した場合には速やかにエリートの座を譲るからである。

4.2 n-Queens 問題

4.2.1 n-Queens 問題の特徴

n-Queens 問題とは、図 4.3に示すように、$n \times n$ の盤 (本来は 8×8 のチェス盤) の上に、n 個の駒 (Queen) を次の条件を満たすように配置する問題である。評価関数の値域が狭いことや、進化の途中の遺伝子列があまり意味をもたないことから、単純な遺伝的アルゴリズムで解くのにはあまり適さないことが予想されるが、組合せ問題として有名なので解いてみることにした。

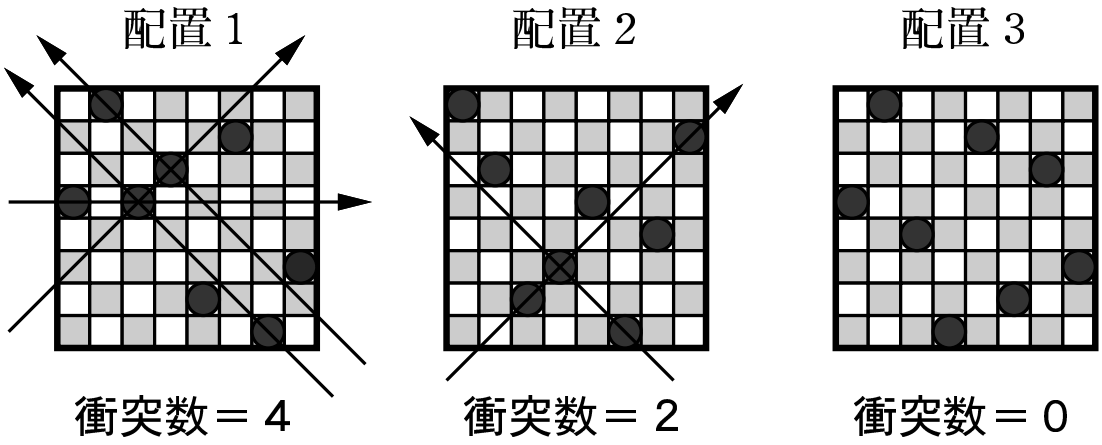


図 4.3: n-Queens 問題 (図は n=8 の場合)

- 条件[1]
  - どの列にも 2 つ以上の駒があってはならない
  - どの行にも 2 つ以上の駒があってはならない
  - どの斜めの直線上にも 2 つ以上の駒があってはならない
  - n 個の駒を必ず全部使わなくてはならない

この問題を遺伝的アルゴリズムを用いて解くために、衝突が発生した直線 (行, 列, 斜め) の本数に比例した評価関数を用いた。

ここでは、n=16 として、16-Queens 問題を解くことにした。染色体の表現は、各遺伝子座を盤上の列に対応させた。遺伝子は 16 値をとり、その値を駒が配置される行に対応させた。この 2 つの情報により、駒が配置される位置が決まる。

各遺伝子座を列に対応させることにより、列方向の衝突は考えずに済むことになり、衝突の回数を減らすことができる。

[1] この条件は元来、チェス盤上に 8 個のクイーンをどのクイーンも他のクイーンをとることができないように配置するための条件である。

4.2.2 結果と考察

結果のグラフを図 4.4に示す。

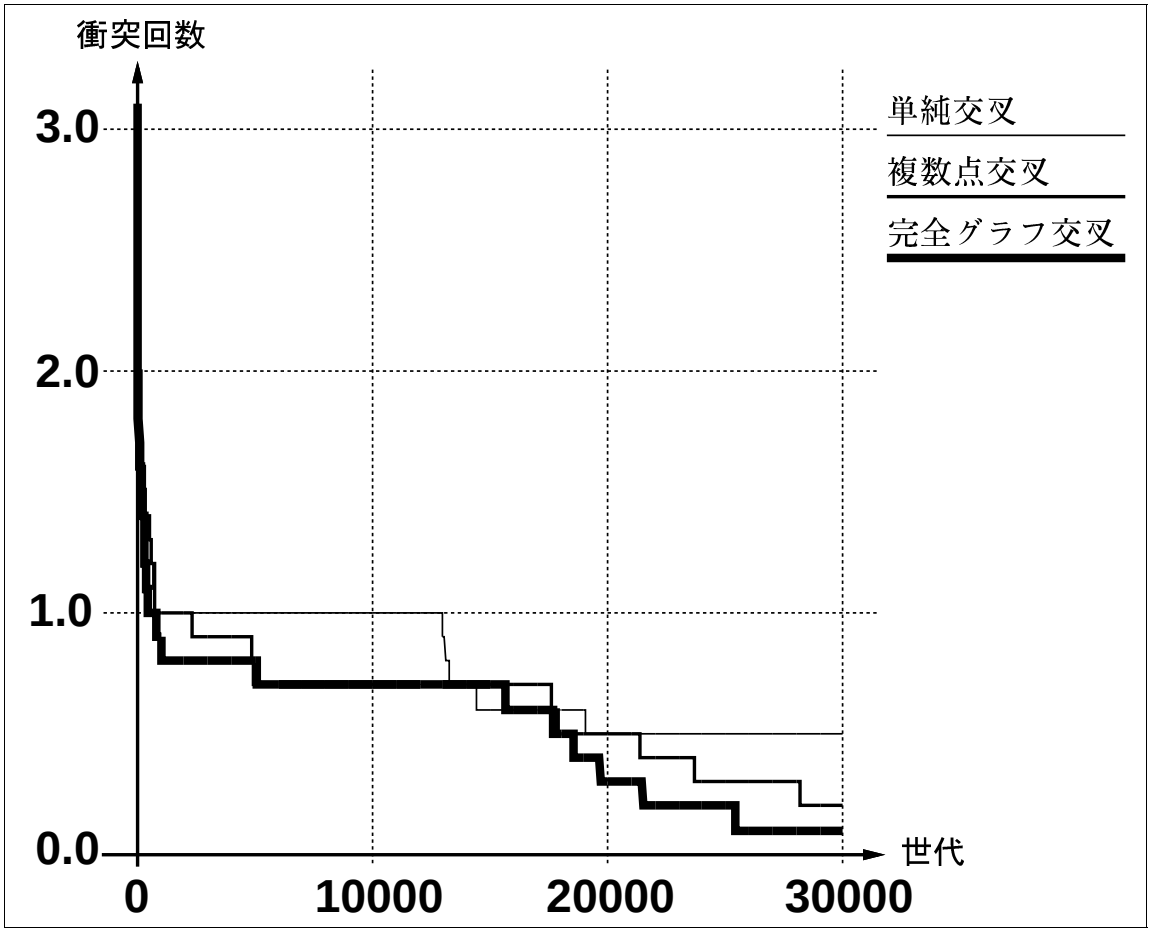


図 4.4: 16-Queens 問題の最良値の推移

この図を見る限り、完全グラフ交叉が他の手法と比較して若干速く進化しているのが分かるが、あまり大きな効果を上げているとはいいがたい。また、処理時間を加味すると、完全グラフ交叉は他の手法よりも進化が遅いことになる。

この結果より、残念ながら、完全グラフ交叉は n-Queens 問題を解くのには適さないといえるだろう。その理由は、n-Queens 問題には多くの解が存在する (ex. n=8 の場合ですら 92 通りある) ので、真の最適値になる遺伝子型が何通りも存在するために、遺伝子座にはあまり依存しない問題となっているからだと思われる。

4.3 ナップザック問題

4.3.1 ナップザック問題の特徴

ナップザック問題とは、図 4.5に示すように、重さと値打ちをもついくつかの物品があり、一定の重さまで収納できるナップザックに、どれだけ値打ちの高いものを収納できるか、というものである。

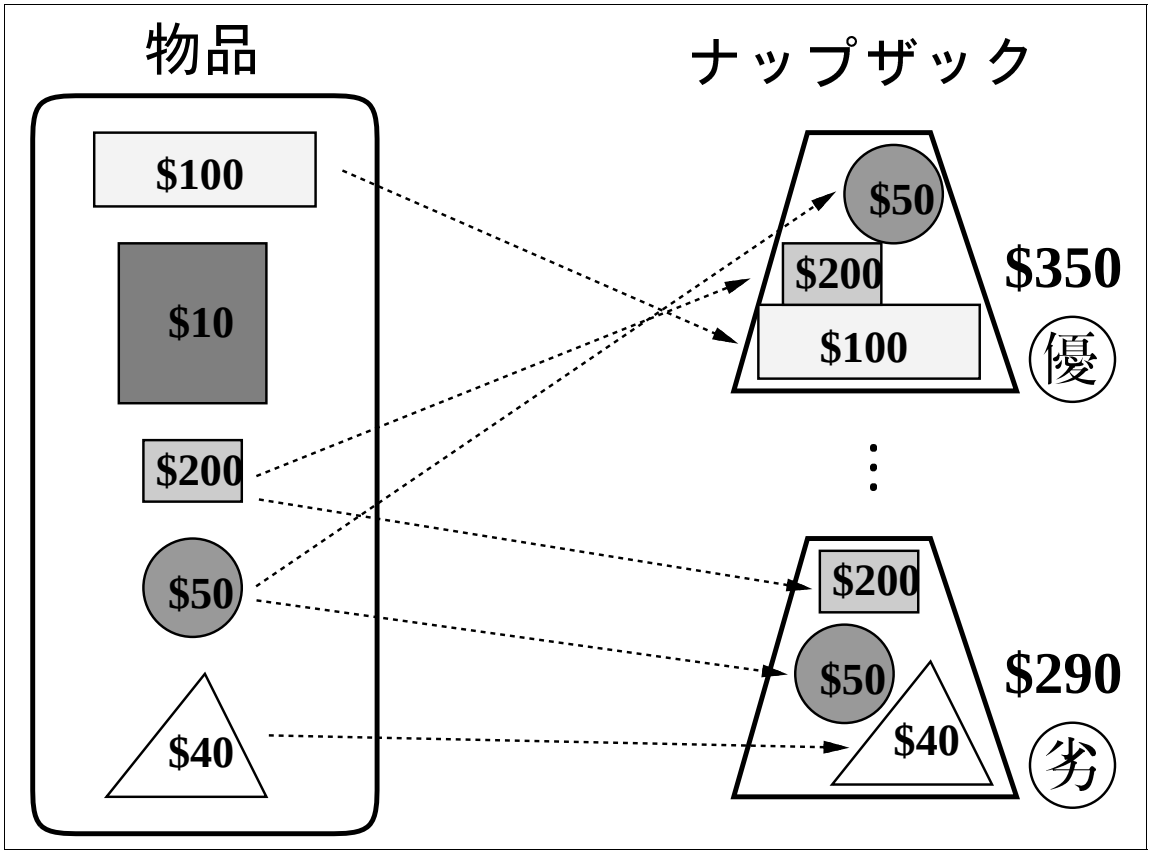


図 4.5: ナップザック問題

4.1で扱った 2 つの例題には局所解は存在しないが、この問題では多くの局所解が存在することが特徴的である。それに加え、遺伝子は「収納する (1)」、「収納しない (0)」の 2 値で済むので、まず最初に評価を簡単に行うには適当だと思われる。

評価関数としては、ナップザックに収納できた物品の価打ちの和を直接用いた。(ただし、ナップザックに収容できる制限重量を超えたときの評価値は、0 とした)

ここで実際に用いた物品は、表 4.2に示した 32 個であり、ナップザックに収納できる重さの上限は 100 とした。一方、全解探索によって真の最適解、493 が得られている。

実験に用いた各定数の値は表 4.3の通りである。

表 4.2: 用いた物品の値打ちと重さ

#	値打ち	重さ	#	値打ち	重さ	#	値打ち	重さ	#	値打ち	重さ
0	20	9	8	2	8	16	28	7	24	21	7
1	7	8	9	21	7	17	4	5	25	28	5
2	33	9	10	2	8	18	35	7	26	14	7
3	36	9	11	12	4	19	14	7	27	40	3
4	2	7	12	28	7	20	40	5	28	22	10
5	2	7	13	22	10	21	14	7	29	22	10
6	22	10	14	40	3	22	7	8	30	21	4
7	21	2	15	40	5	23	4	3	31	21	2

表 4.3: 各種定数の設定値

定数	値
遺伝子	(0,1)
遺伝子長 (物品の数)	32
個体数	100
世代当たりの交叉回数	50
遺伝子座当たりの突然変異率	0.1%
選択の際の評価値の基準	0.6 × 集団の評価値の平均値

なお、選択の際の評価値の基準を表中の式のようにしたのは、あらかじめ最適解が分かっていないためである。また、このようにすることによって、集団の平均値が上がるにつれ選択の際の閾値が動的に変わる (進化するにつれて、閾値は上昇してゆく) ので、進化を効率的にすることにも貢献すると思われる。

4.3.2 実験の結果と考察

世代数を横軸にとった、最良値の推移を図 4.6に示す。

この結果を見ると、完全グラフ交叉が圧倒的な効果を上げていることが分かる。他の手法より早く進化するということ以上に、10 回の平均ですら 10000 世代以内に真の最適解に収束している。平均をとる前の単体のデータを見ると、2000 世代ですでに真の最適解に収束しているものすらあった。

これは、ナップザック問題が遺伝子座に大きく依存する問題であるため、完全グラフ交叉の威力が発揮されているためだと思われる。

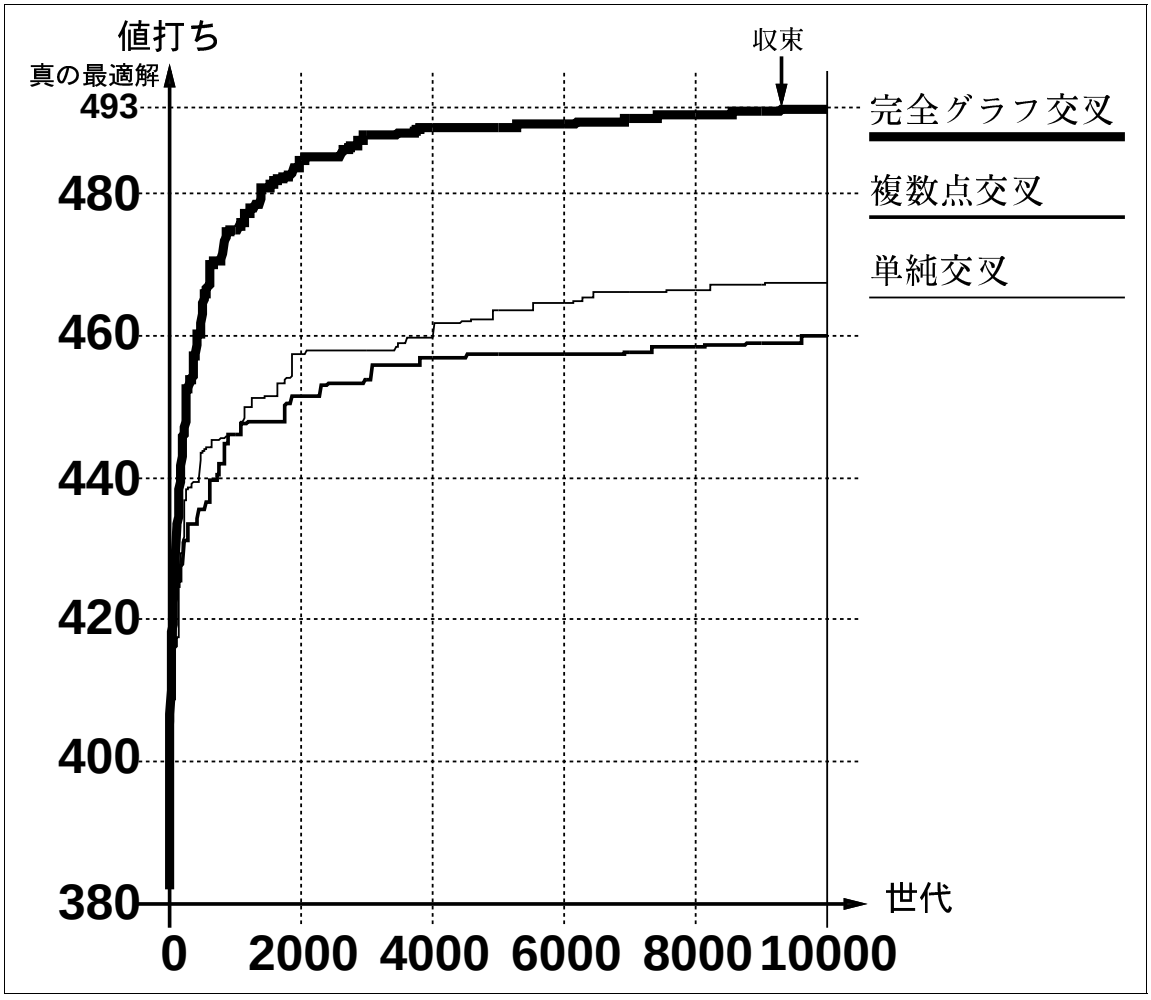


図 4.6: ナップザック問題の最良値の推移

4.3.3 突然変異率に関する実験

2.4.5で述べたが、突然変異は解が局所解に陥らないようにする役割を担っている。ナップザック問題のように、多くの局所解が存在する問題では、真の最適解に至るためには平均的に、突然変異によって隣接する局所解の峰に飛び移れなくてはならない。問題によって局所的な最適解の峰の稠密度は異なるので、問題に応じて異なった突然変異率を採用しなくてはならないと思われる。

そこで、ここでは表 4.2に示した物品の集合に対して、どのような突然変異率が適した値であるかを調べてみた。

なお、ここで示す突然変異率 (MR) の数値は、各世代に各染色体の各遺伝子座が突然変異を受ける確率である。また、突然変異率と収束状況の関係を見るのが趣旨なので、完全グラフ交叉の最良値の推移のみを示す。

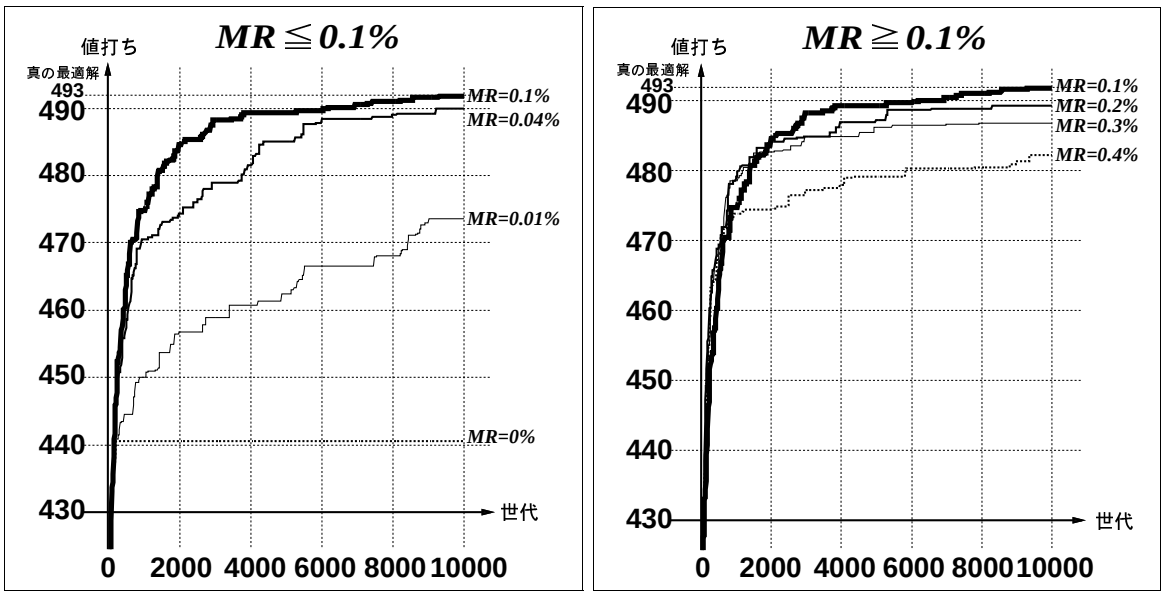


図 4.7: ナップザック問題の最良値の推移 (MR=突然変異率)

見やすくするために、左側は $MR \leq 0.1\%$、右側は $MR \geq 0.1\%$としてグラフを 2 つに分けた。

まず、左側のグラフでは MR=0%で局所解に陥ったまま抜け出せないでいるのが分かる。そして、突然変異率が上昇するにつれ、よい性能を示すことが分かる。

一方、右側のグラフでは、逆に突然変異率が 0.1%を越えると性能の劣化が始まる。

つまり、この例題では 0.1%という突然変異率が適していたことが分かった (正確な最適値は分からないが、およそ 0.1%であろう)。4.3.1の実験で、突然変異率を 0.1%という値に設定していたのは、単なる偶然である。

ここには具体的に示さないが、用いる物品の集合を異なるものに変えて実験してみた結果、若干異なる突然変異率が最適値であることが分かった。

4.3.4 結合係数計算の間引きによる高速化

さて、以上の実験から完全グラフ交叉は少なくともナップザック問題においてはある程度有効な手法であることが分かったが、今までは世代数を横軸にとったグラフしか示してこなかった。アルゴリズムの原理から考えても大幅に処理時間がかかることは予期されていたのだが、実際には (4.3.1のナップザック問題の実験では) 単純交叉の 9 倍、複数点交叉の 8 倍程度の処理時間がかかっている。

たとえ世代数でみて完全グラフ交叉が優れていても、実時間で比較すると性能がよいといえないのでは仕方ない。そこで、時間がかかる処理を高速化できないかと考えた結果浮かんだのが、結合係数計算の省略である (図 3.3)。省略といっても、完全に省くわけではなく、n 世代に 1 回という低頻度で評価を行うのである。高速化を図る一方で、性能を劣化させない程度の n の値を求めてみると、図 4.8と表 4.4のようになり、n は 100 程度で十分であることが分かる。それ以上 n を大きくしても速度の向上が図れないのは、結合係数計算の省略による高速化の限界であるからである。さらに高速化を望むなら、別の処理の高速化を図らなければならない。

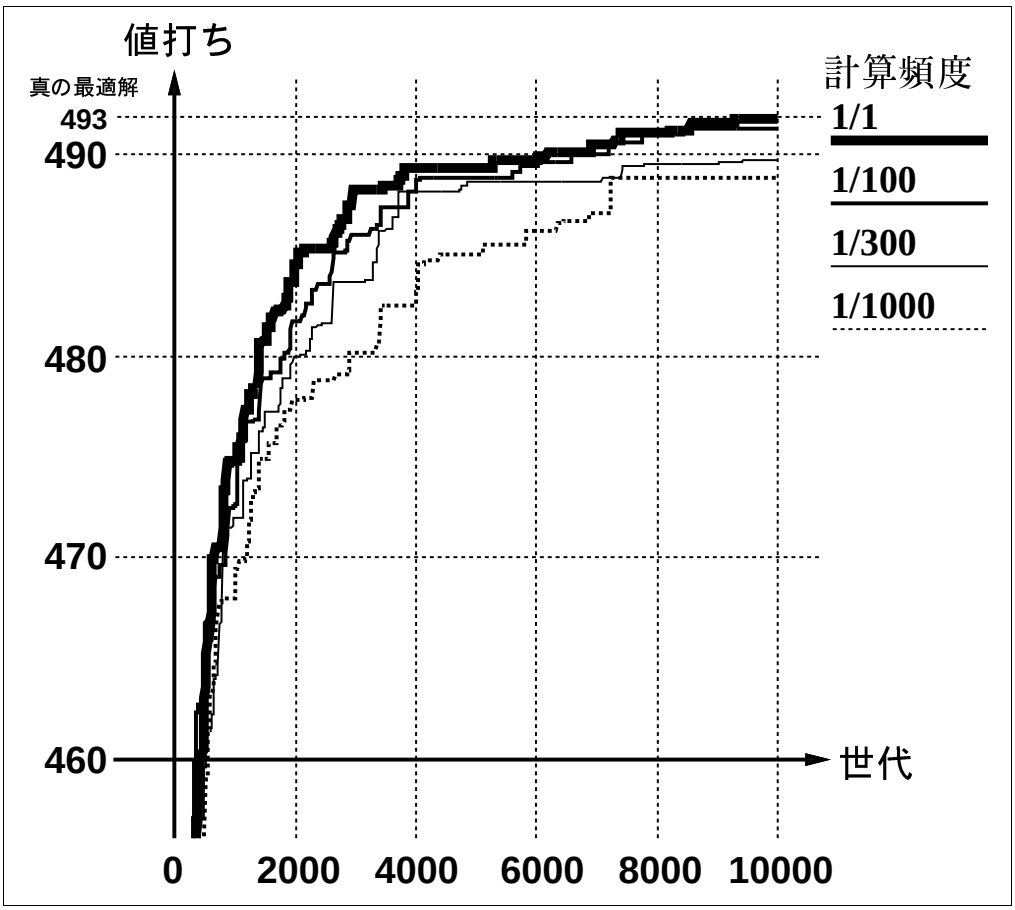


図 4.8: 結合係数の計算頻度と最良値の推移

表 4.4: 結合係数の計算頻度と処理時間 (単純交叉で毎回評価したときの時間を 1 とする)

方式	1/1	1/3	1/10	1/30	1/100	1/300	1/1000	1/3000
単純交叉	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
複数点交叉	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
完全グラフ交叉	9.2	7.7	5.6	5.4	5.2	5.2	5.4	5.3

さて、以上のように結合係数の計算を 100 世代に 1 度にして高速化したものと、他の手法との比較を、実時間を横軸にして図 4.9に示す。

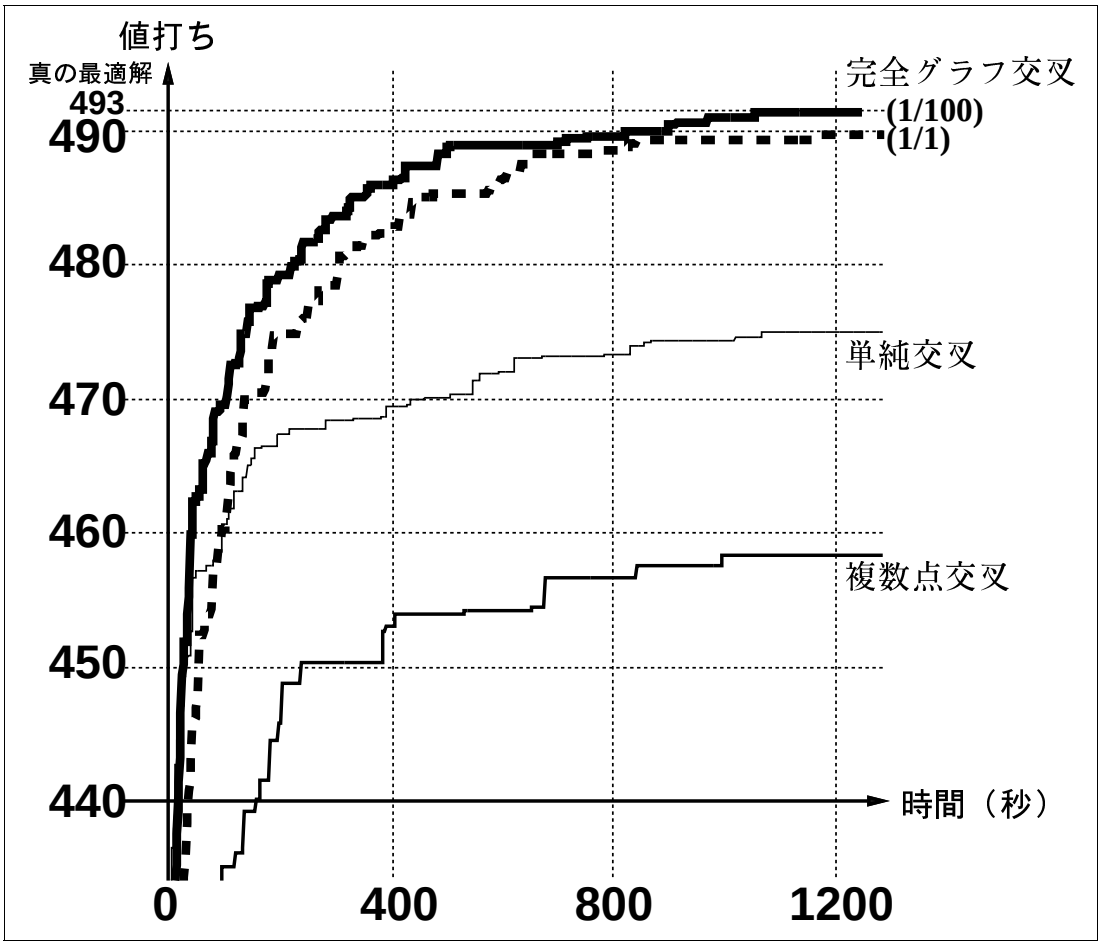


図 4.9: ナップザック問題の実験結果 (最良値の推移)

この結果を見ると、実時間で見ても完全グラフ交叉がよい性能を上げていることが分かる。結合係数の計算頻度を変えて高速化を図った割には、グラフの差が少ないようにみえるが、実際に最適解に達するまでの時間は大きく異なっている。

第 5 章

結論と考察

いくつかの例題を通じて完全グラフ交叉の性能を評価してきた。その結果、完全グラフ交叉は、必ずしも万能ではないが、特定の遺伝子座が評価関数に大きく影響を及ぼすような問題 (ex. 例題 2, ナップザック問題) に対して特に有効だということが示された。

また、世代数でみる限り、どの例題においても、完全グラフ交叉は他の手法と比較して効率的な進化をみせている。

しかし、時間軸でみると、特定の遺伝子座が評価関数にあまり影響を及ぼさないような問題 (ex. 例題 1, n-Queens 問題) では、威力を発揮できなかった。完全グラフ交叉は元来、生物の遺伝子にならって、各遺伝子座が特定の役割を担い、その役割に応じた重要度をもっていると仮定しているので、このような問題に対して効果を発揮しないことは予測されたことではある。

突然変異率については当初、熟慮せずに 0.1%に設定していたのだが、この値も収束状況に影響することが分かった。また、世代が若い間は、突然変異率が大きい方が進化が速く、ある程度収束してくると小さい方がよいことが実験の結果分かっているので、このことを利用して、動的に突然変異率を変化させて進化を効率的にすることも可能だろう。

n-Queens 問題では、遺伝子が 2 値ではなく、多値をとる問題に関してもシステムとして対応できることが示された。遺伝子が純粋な連続値をとることはアルゴリズム上できないが、メモリが許す範囲で実数を標本化すれば、本アルゴリズムを適用することができる。

本論文には記していないが、標準的なベンチマーク問題である、巡回セールスマン問題 (TSP) への適用を考え、文献 [2] に従って致死遺伝子が生じないような工夫 (サブツアー交叉と呼ばれる特殊な交叉を行う) を施したシステムを構築した。しかし後に、そこへは完全グラフ交叉を適用する余地のないことが判明したために、実験をとりやめた。巡回セールスマン問題は、問題自体がすでに完全グラフと類似した性質をもっており、都市間の距離を勝手に変えるわけにはいかない。そのために完全グラフ交叉の特徴である結合係数の概念を用いることができず、完全グラフ交叉を巡回セールスマン問題に適用するのは困難であり、かつ有効でないという結論に至った。

今後の課題としては、結合係数計算の間引き以外の工夫をして、さらなる高速化を図るとともに、この時点で比較している手法はあくまでも最も原始的な単純交叉と複数点交叉のみであるが、実際には様々な工夫が試みられて成果を上げているので、他の有効な手法との比較が考えられる。

謝辞

本論文の研究を進めるにあたって、田中英彦教授及び小池帆平講師からは素晴らしい研究環境を提供して頂いたばかりでなく、御多忙の中を幾度も相談会を開いて頂き、貴重な助言や示唆を与えて下さったことに深く感謝の意を表します。

また、研究を直接指導して頂いた毛利隆夫氏には研究内容の詳細に至るまで指導して頂いたばかりでなく、これからの研究生活の指針となる、研究の進め方をも丁寧に教えて頂き、心より感謝致して居ります。

また、研究室のその他の方々にも、研究の進め方のみならず、日常生活の面でも様々な事柄について教えて頂き、大変お世話になりました。

そしてこの場を借りて、本論文に限らず大学生活全般にわたって、この 2 年間を共に過ごした電気・電子工学科の友人達、4 年間の大学での生活を共に過ごした友人達、様々な面で支えてくれた家族の皆にも深く感謝の意を表します。

ここに挙げさせて頂いた方々のみならず、多くの方々に支えられてこそ、本論文を提出することができたのだと思います。最後に改めて全員に、衷心より感謝の意を表します。

参考文献

[1] 北野 宏明 編,「遺伝的アルゴリズム」, 産業図書, 1993.

[2] 山村 雅幸, 小野 貴久, 小林 重信,「形質の遺伝を重視した遺伝的アルゴリズムに基づく巡回セールスマン問題」, 人工知能学会誌 Vol.7 No.6, 1992.

[3] 米澤 保雄 著,「遺伝的アルゴリズム –進化理論の情報科学–」, 森北出版, 1993.

[4] 南雲 仁一 編, 岩波講座 情報科学 –24「生体における情報処理」, 岩波書店, 1982.

[5] 柴谷 篤弘, 長野 敬, 養老 孟司 編, 講座 進化 –1「進化論とは」, 東京大学出版会, 1991.