平成１０年度知能システム科学専攻修士論文

DNA Computingにおける望ましいDNA配列の設計

的場　琢

Encoding Better DNA Sequence for DNA Computing

Taku Matoba

提出年月日 平成１１年２月１９日

主査教官 小林　重信 教授
審査教官 新田　克己 教授
審査教官 三宅　美博 助教授
審査教官 喜多　一　助教授
審査教官 樺島　祥介 講師

DNA Computing における望ましい DNA 配列の設計

的場 琢

Encoding Better DNA Sequence for DNA Computing

Taku Matoba

Abstract: DNA computing is information processing utilizing DNA molecule. The most critical step in DNA computing is the encoding step that arranges the algorithmic problem onto DNA molecules. If this phase failed, they would attach undesirable points. This "Unintended Hybridization" affects the results as an error. In this paper, we present a new encoding method for DNA computing using genetic algorithms (GAs) on a desktop computer. The evaluation function that is introduced in the leading investigations doesn't fully reflect hybridized condition. Thus, we propose a new evaluation function as well as crossover that corresponds with coding space. This crossover utilizes mutual evaluation with strands. New evaluation function and crossover draw good encodings.

1 はじめに

DNA Computing とは、DNA 分子の物理化学的な性質を利用した情報処理である。この技術は、Adleman が、DNA 分子を利用してハミルトン経路問題を解く方法を提案し、実際に 7 都市のハミルトン経路問題を解くことに成功したことから、急速に脚光を浴びることになったものである [Adleman94]。実際の DNA Computing の手順は、次のように行われる。

1. 塩基配列の決定
2. DNA 断片の生成
3. 反応
   ハイブリダイゼーション
   ライゲーション
4. 結果の表示

まず最初にブール代数の変数やグラフの枝や節点などを１５～２５ (mer) の塩基配列に割り当てていく。塩基配列１つにつき１つの変数や節点を割り当てる。どの塩基配列にどの変数を割り当てるのかについては問題ごとに依存する。ところで塩基配列の１５～２５という長さについては主に２つの理由がある。一つは単鎖の DNA が自分自身と結合するヘアピン構造を取りづらいということ。もう一つは室温の状態で相補的な配列同士が比較的安定して結合することができることである [Adleman94]。

つぎに実際に DNA を生成するわけだが、10^{18} 個程度ならば実験する上できわめて良い精度（結果に悪影響を及ばさない程度）で生成することが可能である。生成する個数は研究によって異なるが、Adleman の実験の場合は長さ 20(mer) の塩基配列を 1 つの変数として、それぞれの塩基配列を 10^{14} 個程度生成する。

3. の反応の作業の中心となるのは、DNA 配列の入った容器を暖めたり、冷やしたりすることである。暖めることによって弱い水素結合で結合していた 2 本鎖の DNA はほどけて単鎖の状態になる。このときに温度を低下させると、DNA はハイブリダイズし、再び 2 重らせん構造を生成する。ここで起こるハイブリダイゼーションという反応は WatsonClick-相補的 (WC-相補的) な関係にある DNA もしくは RNA が結合を行うことを意味する。WC-相補的な関係というのは、次のようなものである。１つ１つの塩基同士は A と T、G と C が結合するが、この A と T、G と C の関係を WC-相補的な関係と呼ぶ。例えば DNA 配列が ATGCCT である場合、相補的な関係の DNA は AGGCAT となる。このとき並びの順番が逆になっていることに注意すること。これは、塩基配列末端には 3'-末端と 5'- 末端があり、相補的な塩基配列は 3' → 5' 方向と 5' → 3' 方向同士でしか結合できないからである (図 2 参照。矢印は 3' → 5' 方向を示す)。ハイブリダイゼーションの他に利用する反応としては、ライゲーションがある。これは、隣接した２つの DNA 鎖を共有結合によって１つの鎖にする反応である[1]。

最後の結果の表示には電気泳動を用いる。

実際に DNA 分子を用いて計算を行う際には、3. で期待されないハイブリダイゼーションが数多く生じ、計算の妨げになることが指摘されている [萩谷 97]。

DNA Computing では、その設計段階に於いて、ある塩基配列とその配列と相補的な関係にある塩基配列を用意する。そして、これらの塩基配列同士が「期待

[1] ここに示した２つの反応の他に、ある特定の配列の部分を切断する酵素や、単鎖の DNA から相補的な DNA を生成する酵素を利用した研究もあるがここでは特に取り上げない。

される」ハイブリダイゼーションを起こすことによって計算が実現される。しかし、実際には塩基配列同士がずれた状態で結合したり、似通った別の塩基配列に結合してしまったりする事がある。このような反応はエラーになる。これを期待されないハイブリダイゼーションと呼ぶ。

この反応が起きる原因はいくつかあるが、[萩谷 97] では最初の段階の塩基配列の決定をランダムに行った事が原因の一つであるとしており、塩基配列の設計の必要性を指摘している。この DNA Computing を行う準備段階の DNA 配列の設計についてはまだその手法は確立されていない。

Garzon らは塩基配列の決定問題を最適化問題としてコンピュータで解決しようとした [Garzon98]。しかし、彼らの研究は、この問題が NP-完全な問題であることを証明し、確率的な探索手法が不可欠であることを指摘した点が注目されるが、問題の規模が大きい場合については、論文で示された塩基配列は最適に達していない。さらに彼らが求めた塩基配列はハイブリダイゼーションを起こす部分的な状況しか反映していない、という問題点もある。

本論文では、従来の評価関数の問題点を改善する新たな評価関数を提案する。改善の主眼となるのは、いかにしてハイブリダイズする可能性のある部位を数え上げるかである。そこで、これらの部位の和を取ることによる情報量の増加を試みた。その結果、Garzon らのグループでは得られていない最適なコードを発見することができた。

また、最適化手法として遺伝的アルゴリズム (GA) を適用する。一般に GA の性能はコード化・交叉に依存することが経験的に知られている [小野 97]。そこで、新たな交叉としてトレーディング交叉を提案する。この交叉は個体内のローカルな塩基配列同士の適応度を考慮した交叉である。その結果、解を求めるのが困難な状況であっても、トレーディング交叉は単純な一様交叉よりも良好な結果が得られることを確認した。

本論文の構成は、以下の通りである。まず第 1 章では DNA Computing の概要と塩基配列の決定問題の必要性について述べた。第 2 章では、この研究で最適化を行う問題を具体的に設定する。第 3 章ではまず、従来手法の評価関数について説明し、その問題点を指摘する。次に、その問題点の解決のために新たな評価関数を提案する。第 4 章では、数多く存在する最適化手法の中でも特にランダムサーチ、マルチスタートローカルサーチ、GA の 3 つについて述べる。第 5 章、ここでは GA の新しい交叉として、トレーディング交叉を提案する。第 6 章では提案手法の妥当性を確かめるために、いくつかの実験を行い、結果について考察を行う。最後の第 7 章はこの論文全体のまとめと成果に関する考察である。

2 問題設定

この論文では特に注釈がない場合、塩基配列といえば長さ $n(\mathrm{mer})$ の塩基配列のことを指す。

2.1 DNA 配列の設計問題

期待されないハイブリダイゼーションをできるだけ起こさないようにするためには、ハイブリダイズする可能性のある状況を調べ上げ、ハイブリダイゼーションが起きないように塩基配列を改善していく手法が有効であると考えられる。計算に用いる塩基配列はあらかじめ決定しておくため、ハイブリダイゼーションが起きる可能性は実際に反応を起こす前に、ある程度の見積もりが可能である。

そこで、この見積もりを評価関数として最適化を行い、塩基配列の改善を試みる。

この決定問題は A(アデニン),T(チミン),G(グアニン),C(シトシン) の 4 種類の記号の並びの最適化問題の一種と考えることができる。ただし、本論文では、エネルギー関数を用いた 3 次元構造の予測までを含んだモデルは考えないものとする。

2.2 nxm 問題について

最適化を行う単位として、長さ $n(\mathrm{mer})$ の断片を m 種類だけ含むものを考える[2]。これらの塩基配列の最適化を行う問題を $(n)\times(m)$ 問題と呼ぶ。例えば 20×10 問題ならば、長さ $20(\mathrm{mer})$ の塩基配列を 10 種類用意する事を意味する。

図 1 は $(n)\times(m)$ 問題で最適化を行った塩基配列をどのように DNA Computing で利用するのかを示している。まず最初に $(n)\times(m)$ 問題を何らかの最適化手法を用いて解く。次に求めた配列に変数やグラフのエッジなどに割り当てていく。そして、これらの塩基配列同士で反応を行い、計算を実行する。$\overline{X1}$や$\overline{X2}$ は、そ

[2] この単位は遺伝的アルゴリズムにおける個体を意味する

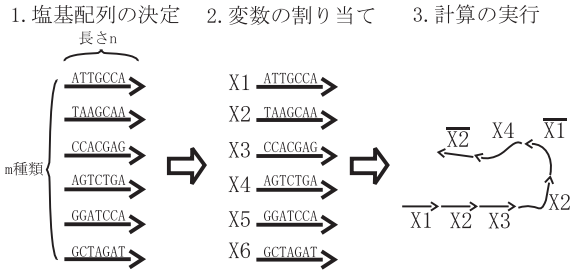

図 1: DNA 配列の設計問題と $n \times m$ 問題との関係

れぞれ $X1$ や $X2$ の相補的な関係の塩基配列である。実際の計算では相補的な関係の塩基配列を利用するのにも関わらず、図 1 でも示したように、m 種類の塩基配列の中には相補的な関係の配列は含めていない。これは相補的な関係の塩基配列は容易に求めることができるので、明示的に示す必要がないからである。

3 評価関数の提案

3.1 Garzon の H-measure

Garzon らが定義した評価関数 H-measure は 2 つの塩基配列がどの程度結合する可能性があるかどうかを評価する関数である。その定義は以下のようになる [Garzon98]。

$$|x, y| := \min_{-n<k<n} H(x, \sigma^k(\bar{y})) \tag{1}$$

x, y はそれぞれ、長さ n の塩基配列である。$\bar{y}$ は y の WC-相補配列を意味する。σ は情報科学におけるシフトを表す演算であり、添え字はいくつシフトするかを示す。$k > 0$ ならば右シフト、$k < 0$ ならば左シフトを意味する。また、$H(*, *)$ は次式で定義される。これはハミング距離と呼ばれているものである。

$$H(u, v) = \sum_{i=1}^{n} \delta(u_i, v_i) \tag{2}$$

ただし

$$\delta(u, v) = \begin{cases} 0, u = v \\ 1, u \neq v \end{cases}$$

ここで u, v は塩基配列、u_i, v_i はそれぞれの塩基配列の i 番目の塩基である。

例えば、u_i と $\bar{v_i}$ を比較して、そのハミング距離が 1 であった場合、2 つの塩基 u_i と v_i は結合しない。逆に、ハミング距離が 0 であった場合には、2 つの塩基は相補的な関係にあり、結合する可能性がある。塩基同士はこのような関係にあるが、塩基配列同士の結合の可能性については、熱力学的な 3 次元構造を考慮する必要がある。ただし、一般的に、H-measure が小さくなるほど、ハイブリダイズする可能性は小さくなると考えられる。また、シフトを考慮しないハミング距離で評価した場合ではあるが、その評価が良好であった場合、DNA Computing の計算に於いてエラーが減少したことが報告されている [Deaton96a]。

H-measure を用いた最適化では、H-measure を最大化するのが目的となる。また、H-measure の上界は n となる。

3.2 H-measure の問題点

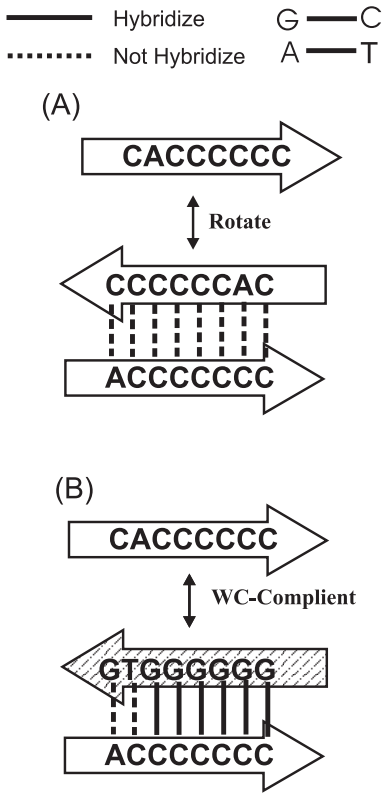

図 2: ハイブリダイズの状態

表 1 は、[Garzon98] にあったものの一部を抽出したものである。任意の 2 つの塩基配列を選び出し、H-measure を適用させると、どのような組み合わせであっても、この評価関数では最適 (最大) になることが見て取れる。

しかし、相補的なものとハイブリダイズすると、とたんに性能が悪くなる。それが、図 2(B) の場合であ

n	Reliable Encoding
8	ACCCCCCC, CACCCCCC, CCACCCCC, CCCACCCC, CCCCACCC, CCCCCACC, CCCCCCAC, CCCCCCCA

表 1: Garzon の最適化済みのコードセット

る。この図では、ハイブリダイズする場合を実線で、しない場合を点線で示した。また、3'→5' 方向を矢印で示している。

これは、このコードセットでは図 2 の (A) の場合だけを想定しているからである。

DNA Computing において相補的な配列を利用しない計算はあり得ない。相補的な配列を含めた評価関数を考える必要がある。

もう一つ問題点は、この評価関数を GA に適用した場合、適応度分布が平坦になってしまうことである。

H-measure を用いた個体の評価では塩基配列の長さを n とすると、H-measure の範囲は $0 \leq |x, y| \leq n$ に限定される。これは、シフトを含むハミング距離において最小の値を取るように定義しているからである。

$n \times m$ 問題の場合、解の候補は、4^{nm} 種類にもなる[3]。それに対し、評価値は n 種類の値しか取れない。結果として個体の適応度分布は非常に平坦なものになる。

平坦な適応度分布は、GA が次に向かうべき適応度を見失い、その性能がランダムサーチ並に低下する原因になる。

3.3 評価関数 H-sum の提案

H-measure の問題点として、ハイブリダイズしている状況の一部しか反映していないことと、適応度地形があまりにも平坦すぎる点を指摘した。本章では、これらの問題を解決するために、H-sum-sum、H-sum-max、H-measure-max を提案する。

[3] 実際に探索を行う解空間はコード化を工夫するなどして、ある程度小さくすることが可能である。しかし、解空間が広大であることに変わりはない。

3.3.1 定義

H-sum を次のように定義する。

$$|x, y| := \sum_{-n<k<n} \{C(x, \sigma^k(\bar{y})) + C(x, \sigma^k(y))\} \quad (3)$$

また C は

$$C(u, v) = \sum_{i=1}^{n} \delta(u_i, v_i) \quad (4)$$

である。ただし

$$\delta(u, v) = \begin{cases} 1, u = v \\ 0, u \neq v \end{cases}$$

である。$H(*, *)$ は、一致しないときに 1 をとるハミング距離であるが、$C(*, *)$ は $H(*, *)$ とは逆で、一致したときに 1 を取る関数である。言い換えれば $C(*, *)$ の値が大きくなるほど塩基配列同士がハイブリダイズする可能性が高くなる。

H-sum を用いた最適化では H-sum を最小化するのが目的となり、評価関数の下界は 0 となる。

次に H-measure を定義し直す。$C(*, *)$ と $H(*, *)$ には、次のような恒等式が成立する。

$$H(x, y) = n - C(x, y) \quad (5)$$

式 (5) が成立するため、H(*,*) で成り立つ性質は C(*,*) でも、ほとんど同じように成り立つ。そのため、H-measure では、実際にはハミング距離を使っているが、以降の議論ではすべて C(*,*) に置き換えることにする。

$$|x, y| := \max_{-n<k<n} C(x, \sigma^k(\bar{y})) \quad (6)$$

H-measure や H-sum は 2 つの塩基配列の間の関係しか定義していないが、m 種類の塩基配列を評価するときには、任意の 2 つの塩基配列すべてに対して評価を行わなければならない。その時、和を取るか、最大を取るかの選択の余地がある。

2 つの塩基配列の H-measure による評価の内で、最大の値を H-measure-max、H-sum を 2 つの塩基配列すべての組み合わせに対して評価し、和をとったものを H-sum-sum、H-sum を 2 つの塩基配列すべての組み合わせに対して評価し、最大を取るものを H-sum-max、と定義する。

また、ここでは分解能も定義する。これは「評価値が離散的な場合にいくつの数値を取りうることができるかできるかを示す個数」である。一般に、この値が大

きくなればなるほど、適応度地形は起伏に富んだものになる。ただし、バリエーションの中のごく限られた値しか取れない場合もあるので、すべての場合に成り立つわけではない事に注意しなければならない。このとき評価値が取りうる値の分解能は、H-sum-max の場合 $O(n^2)$ のオーダー、H-sum-sum の場合 $O(n^2m^2)$ のオーダーである。H-measure-max は $O(n)$ のオーダーであるので、かなり分解能が向上している。

またその他の変更点として、右辺に新たな項が付け加わったことが挙げられる。

式 (3) の $C(x, \sigma^k(\bar{y}))$ は、図 2 (A) の状態を反映したもので、これは H-measure で定義したものとと、ほぼ同じものである。$C(x, \sigma^k(y))$ は、図 2(B) の状態を反映したものである。(B) の状態を反映しているとはいえ、関数だけを見ると、あたかも 2 つの関数の類似度をシフトしながら評価しているように見える。

3.3.2 性質

新たに定義した評価関数の性質について述べる。第一に、3 つの評価関数がどのような側面を重視したのかについて述べる。

H-measure-max この評価関数を用いた最適化では、すべての塩基配列が H-measure の値以下になることが保証される。しかし、3.2 で述べたような問題点が存在する。

H-sum-max この評価関数は、できるだけ 2 つの塩基配列のハイブリダイズする情報を集めようとする評価関数である。しかし、特定のシフトの状態に悪い評価が集中する可能性がある。ただし、H-sum-sum のように、特定の塩基配列同士に悪い評価が集中するような事態は起きない。

H-sum-sum この評価関数は、H-sum-max より多くのハイブリダイズの情報を集める評価関数である。すべてのシフトを考慮した塩基配列の比較に於いてハイブリダイズする可能性のある塩基の部位を数え上げる。特定の塩基配列同士に悪い評価が集中する可能性がある。

H-sum-max と H-sum-sum の利点は、分解能の向上による最適化性能の向上であるが、分解能の向上と引き替えに、評価の悪い収束解に収束する可能性が生じるという問題点がある。

第二に、三つの評価関数に対して成り立つ不等式について述べる。任意の塩基配列に対して次の不等式が成り立つ。

$$H - sum - sum \geq H - sum - max \geq H - measure \tag{7}$$

一般に 3 つの評価の大きさを比べることはあまり意味がないが、この不等式と同じような大小関係が適応度地形に関して、一部の例外を除くものの存在すると考えられる。また、この式から、H-sum-sum で 0(最適) に収束したコードセットは、H-sum-max、H-sum-sum でも 0 に収束することが分かる。

最後に、H-measure、H-sum、2 つの評価関数に共通の性質として可換性をあげることができる。

$$|x, y| = |y, x| \tag{8}$$

3.3.3 その他の結合の状態

ハイブリダイズする状態として、図 2(A)、図 2(B) の他に次の状態が考えられる。

(C) 2 つの塩基配列がともに相補的な配列の場合

(D)(B) とは逆に塩基配列 x が相補的な関係にある場合

状態 (C) については、x と y がともに相補的な配列の場合、結局 (A) と同じ評価になる。状態 (D) は、先に述べた可換性から明らかに (B) と同じになる。式で表すと、

$$C(x, \sigma^k(\bar{y})) = C(\bar{x}, \sigma^k(y)) \tag{9}$$

$$C(x, \sigma^k(y)) = C(\bar{x}, \sigma^k(\bar{y})) \tag{10}$$

となる。以上の議論から式 (3) で定義した H-sum を評価すれば十分であることが分かる。

4 最適化手法の検討

この章では、GA を適用する妥当性の検討のため、いくつかの最適化手法との比較を行う。提案する手法は、ランダムサーチ、マルチスタートローカルサーチ、GA の 3 つである。

4.1 コード化

最適化手法の検討の前に、最適化を行うためのコード化を行う。DNA 断片の長さ n(mer)、塩基配列の数

m の問題を考える。このとき、$n \times m$ の配列を１つの個体とする。この配列 M には A,T,G,C の４種類の文字列が含まれている。また、行列 M の i 行 j 列成分を、m_{ij} とする。

4.2 ランダムサーチ

この問題におけるランダムサーチは、

1. ランダムに解候補を作る
2. いままでの最良と比較して良ければ交換、悪ければ棄てる

を繰り返すものである。

4.3 マルチスタートローカルサーチ

一般に、ローカルサーチとは「ひとつしか個体を使わず、その近傍の探索を繰り返す」もので、山登り法とは、ローカルサーチかつ「解が改善されるときに限って個体を更新する」ものである。ただし、ここでは山登り法の意味でローカルサーチという言葉を使うものとする。

マルチスタートローカルサーチとは、ローカルサーチ（山登り法）とランダムサーチの組合せで、ローカルサーチの初期解をランダムに生成して、複数回のローカルサーチを実施するものである。

ローカルサーチのオペレータについては、それぞれの解候補 (個体) 単位で行うものとする。具体的には、次のようなオペレータを適用する。

個体内交換 ランダムに選択した m_{ij} 要素と m_{kl} 要素を交換する

突然変異 ランダムに 1 つの m_{ij} 要素を選び出しその要素を他の塩基に変化させる

個体内の 1 個体に個体内交換と突然変異を順番に適用する。また、この操作をすべての個体に行った時点で 1 つ世代を進め、再びすべての個体にオペレータを適用していく。

4.4 GA の枠組み

GA の操作は個体を単位として行われる。個体が集まったものを、集団と呼ぶ。個体の内部表現を染色体と呼ぶ。個体には、表現型 (phenotype) と遺伝子型 (genotype) の 2 つの側面がある。表現型とは問題側の解空間における個体の表現である。これに対して、遺伝子型とは染色体空間における個体の表現である [小野 97]。交叉や突然変異のオペレーションは遺伝子型に対して行われるが、問題のクラスによっては、表現型を考慮したオペレーションを考えなければならない場合もある。

GA の手順は、初期集団の生成、複製選択、子の生成、生存選択の 4 つからなる [佐藤 97]。

1. **初期集団の生成** 初期集団となる複数の個体をランダムに生成し、各個体を評価する
2. **複製選択** 個体の複製のために、現世題の集団から親となる個体を選択し、親集団へコピーする
3. **子の生成** 親集団に、交叉・突然変異を適用して子集団を生成し、評価する
4. **生存選択** 現世代の集団と子集団より次世代に生存する個体を選択し、次世代の集団へコピーする

GA が他の 2 手法と大きく異なる点は、世代交代モデルによる生存競争の導入、個体同士の交叉である。これらの手法を導入することによって、他手法では乗り越えることのできない適応度分布を乗り越えることが可能になる。

5 トレーディング交叉の提案

交叉はその設計に成功すれば、最適解に収束するために必要な優良スキーマ、ビルディングブロックを蓄積する。そのため、非常に強力な探索オペレータになる可能性を秘めている。

しかし、その設計に失敗した場合、その交叉はビルディングブロックを破壊する方向に働き、ランダムサーチとあまり変わらない性能になってしまうことが経験的に知られている [山村 94]。

この章では、新たな交叉オペレータとしてトレーディング交叉を提案する。

5.1 交叉の原理

交叉オペレータに要求されるのは

継承すべき形質を維持できるように設計すること

である。

ここでいう形質とは１個１個の個体の個性のことを指している。また、形質はコード化されたデータ構造に依存しない、より問題の性質に根ざした特徴である。継承すべき形質とは、その特徴を残しておくことが、評価関数の改善に良い影響を与えることが多いと考えられる特徴を指している。

DNA 配列の決定問題では、継承すべき形質は塩基配列そのものではなく、２つの塩基配列同士の関係であると思われる。

しかし、この形質は、配列が変化すればめまぐるしく変化するし、他の個体に組み込まれた場合には、再評価をしなくてはならない。ある個体では、どうしようもないと思われていた配列が、別の個体では、優秀な評価を受ける可能性があるからである。さらに、配列間の評価と、個体全体の評価が密接に関係しているために、１つの配列で起こった変化が、個体全体の評価に大きな影響を及ぼすことも考慮すべきである。この問題は問題の規模が大きくなればなるほど深刻なものになる。

以上の点から、形質をできるだけ継承するためには、交叉を塩基配列単位で行うことが必要である。交叉をこれ以下の単位で行う場合、配列同士のつながりを調べるために評価関数の再評価が必要になり、計算コストが爆発的に増加してしまう。

配列の途中で切断し張り合わせるような１、２点交叉そして一様交叉は、このコード化のもとではむしろ突然変異的な作用を及ぼすと考えられる。

図 3 にトレーディング交叉の原理を示す。塩基配列 A と塩基配列 B を考えるとき、$A \to B$ と $A \leftarrow B$ の評価値は互いに等しくなる。また、評価ではすべての塩基配列同士の組み合わせを調べることから、評価値の関係は完全無向グラフで表される。このグラフを評価値の高い部分でカットしてやると、全体の評価に悪影響を及ぼす配列同士を分離することが可能になる。こうして２つに分離した個体を互いに交叉させる。

問題はどのようにして、分離を行うかである。

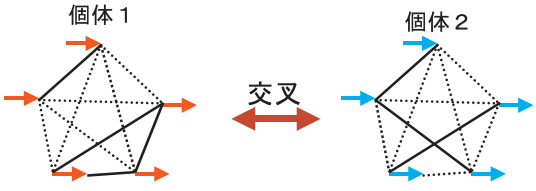

図 3: トレーディング交叉の概要

5.2 交叉の近似的実現

計算時間を問題にしなければ、良い評価の枝だけを残して分離するアルゴリズムを考えることは可能だが、計算コストがかかることと、交叉のバリエーションがほとんどない点が問題である。

そこで、最良の枝ではないものの、ある程度良い評価の枝を残し、かつ確率的にいくつかのバリエーションがある交叉を提案する。

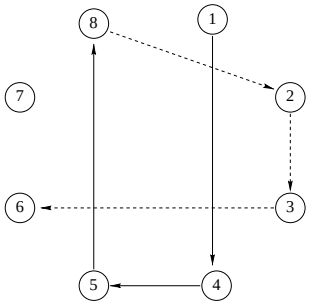

図 4: 近似的トレーディング交叉の原理

図 4 は、近似的トレーディング交叉の概要について説明した図である。①から⑧までの番号の節点は、DNA の配列を示している。つまりこの図は、$(n) \times 8$ 問題の場合を示している。実際には、これらの節点はすべて H-sum などの評価関数で評価した重みのノードによって、完全グラフを構成しているが、図が煩雑になるため、これらのノードについては省略している。

図中、実線で示した経路が近似的トレーディング交叉で利用する経路である。この経路は次のようにして求める。

1. ランダムに任意の節点を選択する。図の場合では①の節点が選択されている。
2. 次に向かう経路を決定する。このとき①から出ているノードの重みを評価し、評価値の最も良好な経路を選択する。この場合、④の節点を選択した。
3. 以下同様に経路を選択して行くが、一度通った節点は選択してはならない。
4. 節点のうちの半分を選択するまで経路を探索する[4]。

結果として $1 \to 4 \to 5 \to 8$ の経路が選択された。この①、④、⑤、⑧の 4 つの節点は個体に残り、残りの②、③、⑥、⑦が、トレードに出され、同じ評価を行った

[4] 一般に m 種類の塩基配列がある場合、m が偶数の場合 $m/2$、m が奇数の場合 $((m-1)/2)$ 箇所の節点を結ぶまで節点を選択して行く。

もう１つの個体との間で交叉が成立する。

このようにして決定した経路は、図 3 のような最適な部分配列集団ではないが、ある程度良い評価を与えられる集団である。より具体的に言えば、この経路は、それぞれの節点について最良の経路と 2 番目、あるいはそれに準ずる程度の経路を選択していることになる。

この経路の探索では、最初にランダムに節点を選び、後は、決定的に選択していくため、m 通りの選択肢がある。①から出発した経路が⑧で終了しているが、逆に⑧が選択されたときに実線の矢印を逆にたどっていくとは限らない。点線のような経路が選択される場合もあるのである。

6 実験

6.1 コード化・世代交代モデル

以下の実験においては、$(n) \times (n)$ 問題の最適化をコンピューター上で行う。また、コード化・世代交代モデルについてはすべての実験で共通であるため、ここで定義する。

コード化は次のように行う。まず、A,T,G,C ４種類の値をとる配列を n 個確保する。この長さ n の配列を、m 個集めた n 行 m 列の２次元配列を１つの個体とする。

また、GA の世代交代モデルについては MGG [佐藤 97] を用いる。この最適化問題においては似通った個体同士の交叉は評価値の低い子が生成される場合が多く、こうした個体同士の交叉はなるべく避けるべきである。そのためにバラエティに富んだ個体を生成することが求められる。そこで、多様性維持に優れている MGG を用いる。

6.2 予備実験

本実験では、ランダムサーチ、マルチスタートローカルサーチ、GA の 3 つの最適化手法の性能比較を行う。

6.2.1 実験条件

GA のパラメータは以下の通りである。

交叉 交叉は一様交叉を用いる。交叉は個体１の m_{ij} 成分と個体２の m_{ij} 成分との間で行い、2 つの個体のすべての成分で適用する。

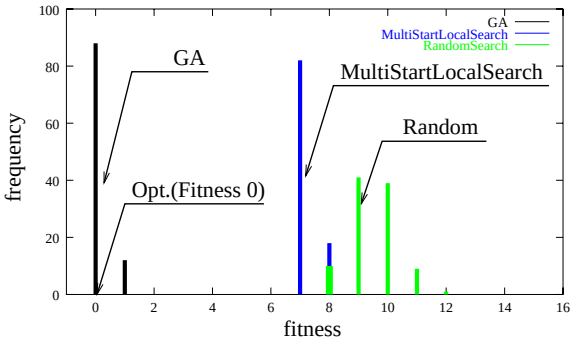

図 5: 3 つの手法の頻度分布 (16x16 問題、100 試行)

突然変異 突然変異はあらかじめ決定しておいた変異率（１％）に基づいて、１つ１つの塩基に対して行った。

評価関数 評価関数は H-measure-max をもちいる。H-measure は

$$|x, y| := \max_{-n<k<n} C(x, \sigma^k(\bar{y})) \qquad (11)$$

である。

その他のパラメータ 個体数 100、子個体の生成数を 10 とした。打ち切り世代数は、50000 とした。

マルチスタートローカルサーチでは、50000 世代ローカルサーチを行った。個体数は 100 とした。ローカルサーチのオペレータは、4.3 で定義した個体内交換と突然変異を採用した。

ランダムサーチでは、他の 2 手法との比較のために (打ち切り世代数)$\times$ (GA の子個体の生成数) の解候補、つまり 5.0×10^5 個生成した。

6.2.2 実験結果

図 5 に、16×16 問題を最適化した場合のランダム、マルチスタートローカルサーチ、GA の収束解の頻度分布を示す。ランダムとは、ただランダムに 100 個体を生成してその分布を見たものである。ランダムサーチの結果として、表 2 に、5.0×10^5 個解候補をランダムに生成した場合の頻度分布を示す。

適応度	7	8	9	10
頻度	1	30595	269775	164013
適応度	11	12	13	14
頻度	31469	3753	373	21

表 2: ランダムサーチの頻度分布 (500000 試行)

6.2.3 考察

探索性能を比較してみると、探索能力の高いものから順番にGA、マルチスタートローカルサーチ、ランダムサーチの順番になった。また、他手法では最適に達していないのに対し、GAでは、およそ9割の試行で最適に達している点は注目に値する。

表2を見ても分かるようにランダムサーチでは、5.0×10^5 回試行を行ったにもかかわらず、適応度8以上の評価を得た個体は1つのみであった。これは、評価値8と7の間には、ランダムでは乗り越えられない、何らかの困難な状況が存在している可能性を示唆している。また、適応度9と10に解が集中しているが、これは、ランダムな解の生成では解の塩基配列に含まれる塩基の内のおよそ半分がハイブリダイズする可能性があることを意味する。

マルチスタートローカルサーチでは、評価値8はほとんどの個体で乗り越えることができたが、最終的には、評価値7に収束してしまっている。

ローカルサーチの性能は、更にオペレータを改良すれば改善される可能性は存在するが、GAのオペレータが、一様交叉と突然変異のみであり、そのほとんどの試行で最適に収束した事を考慮すると、最適化にローカルサーチではなくGAを用いる妥当性は十分にあると思われる。

6.3 実験1

この実験の目的は、H-measureの最適解を発見することである。また、最小ではなく、和を取る評価関数H-sum-max、H-sum-sumのGAにおける挙動についても調べ、考察を行う。

実験1では図2(A)の場合だけを評価する。

6.3.1 実験条件

交叉 交叉は一様交叉を用いる。交叉は個体1の m_{ij} 成分と個体2の m_{ij} 成分との間で行い、2つの個体のすべての成分で適用する。

突然変異 突然変異は導入せず、交叉のみで実験を行った。これは、できるだけシンプルな実験条件を設定することにより、評価関数の違いによる差異を際だたせるためである。

評価関数 評価関数は、H-measure-max、H-sum-max、H-sum-sum 3つを用いて、それらの性能を比較する。実験1ではH-measureは

$$|x,y| := \max_{-n<k<n} C(x,\sigma^k(\bar{y})) \tag{12}$$

である。またH-sumは

$$|x,y| := \sum_{-n<k<n} \{C(x,\sigma^k(\bar{y}))\} \tag{13}$$

である。

その他のパラメータ すべての実験で個体数100、子個体の生成数を10とした。10000世代以上評価が更新されない場合は計算をうち切った。

6.3.2 実験結果

3種類の評価関数を用いて評価を行った。16×16問題では、H-sum-sumを用いた場合にだけ評価値0の最適値に収束した。3つの評価値の代表的な収束曲線を図6〜8に示す。また、3つの評価関数についてそれぞれ25回ずつ評価した場合の頻度分布を図9〜11に示した。

H-sum-sumが0に収束した場合、式(7)よりH-sum-max、H-measureでも、評価値は0になり、すべての評価関数で最適なコードが発見できたことになる。最適に収束した16x16問題の塩基配列の集合を表3に示す。

塩基の長さが16の場合のコード化は、Garzonらのグループでは最適には達していない。

6.3.3 考察

この実験では、評価関数の和を取ることで、適応度地形の起伏が強調され、結果として16x16問題で確実に最適値に収束する事が可能になった。16x16問題だけではなく、30x30問題まで規模を広げても最適に収束することが、これまでの実験で確認することができている。

[Garzon98]には、追試ができるような計算に用いた細かいパラメータや、探索手法についても詳しくは説明されていないのだが、論文に紹介されている塩基配列の並びから判断すると、H-measureをそのまま評価に利用しているために、初期収束に陥っているのではないか。という印象を受ける。

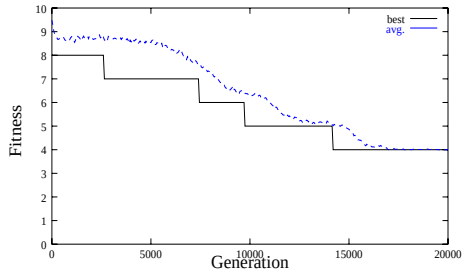

図 6: 16x16 問題における収束曲線 (H-measure)

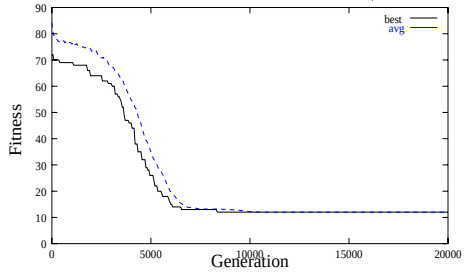

図 7: 16x16 問題における収束曲線 H-sum-max)

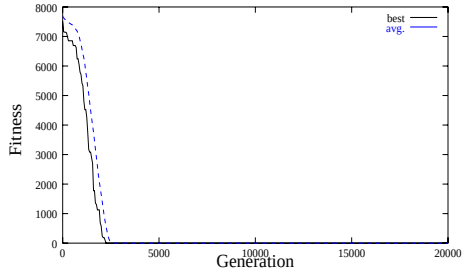

図 8: 16x16 問題における収束曲線 (H-sum-sum)

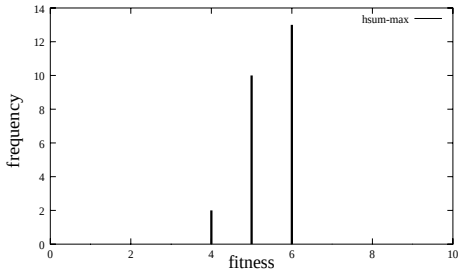

図 9: 最適解の頻度分布 (16x16 問題,H-measure)

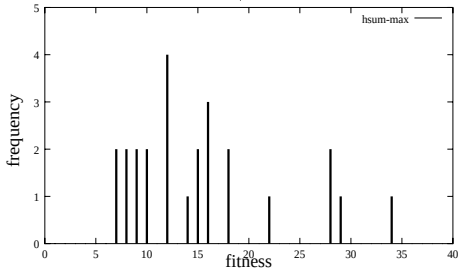

図 10: 最適解の頻度分布 (16x16 問題,H-sum-max)

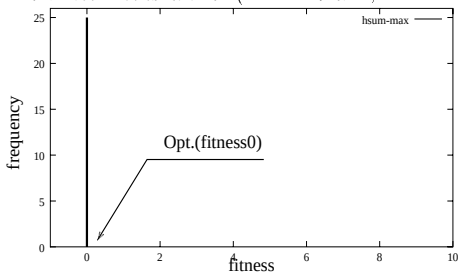

図 11: 最適解の頻度分布 (16x16 問題,H-sum-sum)

今回の実験では、ごく一般的な一様交叉のみで、特に新たな交叉を導入することなく収束している。これは、1つには世代交代モデルの力に依るところがある。

だが、最も有効であったのは、評価関数の和を取ることによる効果であろう。この最適化問題の場合は 2 個体の評価すべての和を取ることで、GA の探索性能が向上し、初期収束に陥ることなく最適値にまで収束することが可能になることが分かった。

6.4 実験 2

トレーディング交叉の性能を比較するため以下のような実験を行った。

6.4.1 実験条件

交叉 トレーディング交叉と一様交叉の両方で評価を行い両者を比較する。

突然変異 交叉の性能を見るため突然変異は導入せず、交叉のみで評価を行う。

評価関数 評価関数には、H-measure を用いる。

$$|x, y| := \max_{-n<k<n} C(x, \sigma^k(\bar{y})) \qquad (14)$$

最適化のために式 (14) を利用した H-measure-max を用いる。H-measure-max を使用するため、適応度分布は平坦なものになり一般に最適解を探し出すのは困難になる。

その他のパラメータ 個体数は 200、子個体の生成数を 10 とした。

6.4.2 実験結果

トレーディング交叉と一様交叉の場合について性能を比較した。問題は 20×20 を用いた。代表的な収束曲線を図 12 に示す。また、2 つの交叉を利用して得られ

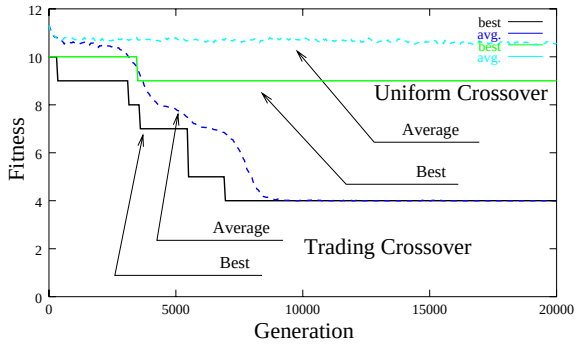

図 12: 収束曲線 (20x20 問題)

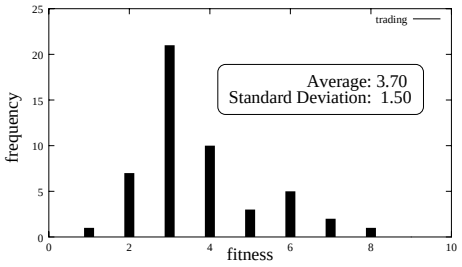

図 13: 収束解の頻度分布 (トレーディング交叉)

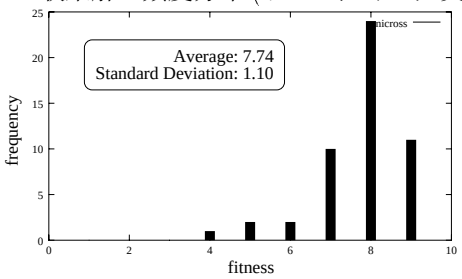

図 14: 収束解の頻度分布 (一様交叉)

た収束解をそれぞれ 50 回ずつ評価した場合の頻度分布を図 13、図 14 に示す。

6.4.3 考察

図 12 を見ても分かるように、トレーディング交叉は一様交叉と比較して、良好な適応度を得ていることが分かる。また、収束解の頻度分布を見ても、分布の平均はトレーディング交叉を用いた方がより良い値を示している。

6.5 実験 3

相補配列を考慮した評価関数の性質とトレーディング交叉の有効性を確かめるために以下の実験を行った。

n	Reliable Encoding
16	TTTCTCTTCCTCCCCC,
	CCTCCTCTTTCTTTCC,
	CTCTTTTTTCCCTCCC,
	CTCCTCCTCCCCCTCT,
	TTTCCCCCCCCCCCTC,
	CTTCTCCCTCCCTTCT,
	CTCTCCCTCTCCCCTC,
	TTCTCCTCTTCTTCCT,
	CTTTTCCTCCCCCCCT,
	TCCCTCCCTTTCCTCC,
	CTCCCTCCTTCTCTTC,
	CTTCTCCTCCTTCTCC,
	TCTTTTCTTTTCCCCT,
	TTTCTTCCCTCTCCCT,
	TTCCTTCCTCCCTCCT,
	CTTTTCCCCCCTTTCC

表 3: 16x16 問題における最適なコードセット

6.5.1 実験条件

交叉 交叉は一様交叉とトレーディング交叉で性能を比較する。

突然変異 突然変異はあらかじめ決定しておいた変異率に基づいて、1つ1つの塩基に対して行った。

評価関数 評価関数は、実験 1 で良好な結果を残した H-sum-sum を使う。また、H-sum は式 (3) に基づく。

その他のパラメータ この実験では、個体数 200、子個体の生成数 10、突然変異率は 1% である。また、10000 世代以上評価が更新されない場合は、これ以上更新されることはないと考え、計算を打ち切った。

6.5.2 実験結果

20×20 問題で最適化を行った際に収束した解を表 4 に示す。

トレーディング交叉、一様交叉、ともに 20 回づつ評価を行ったが、すべての回で、同じ評価値 (36000) に収束した。

また、図 15 では、20×30 問題における代表的な収束曲線を示す。この場合にも 20 試行行った結果、ト

n	Reliable Encoding
20	TATAATATTAATTAAATTAA,
	GGCGGCGCGCGGCCGGCGCC,
	GCCGGGCCGCGGCCGGCCGG,
	TTTAATATTAATTAATTAAA,
	CCCGCCGGCGGGCCCGGCGG,
	TTAATAAATAAATTAAAAAT,
	CGCGCCCCCCGCCCCGGGCC,
	TATATTAATAATTAATTTAA,
	ATTAAAATTAATTATTTAAA,
	CGCCCCCCGCCGGCGGGGGG,
	ATAAATATTAATTAATTAAA,
	CCGGGGGGGCGGGCCCCGCC,
	TTTAATATTAAATAATTTAA,
	GGGGGGGCGCGGGCCCGGCG,
	TTTAATTTTAATTAATTTAT,
	GGCGCCCCCCGCCGGGGGCC,
	TTTAATATTAAATAATTAAA,
	GCCGGGCGGCGGCCGGCCGG,
	TTTAATTTTAATTAATTAAA,
	GGGGGGGCGCGGCCGGGGGG

表 4: 20×20 問題におけるコードセット

レーディング交叉、一様交叉の両方の場合についてすべての同じ評価値 (84000) に収束した。

収束の性質を見てみると、トレーディング交叉は 7000 世代周辺で収束し、一様交叉は、17000 世代近くで収束している。トレーディング交叉は一様交叉に比べて、世代数の比較ではおよそ 2 倍早い世代数で収束している事になる。実際の計算時間はトレーディング交叉の評価が入る分、世代数ほどの差にはならないが、それでもトレーディング交叉のほうが早く収束する。

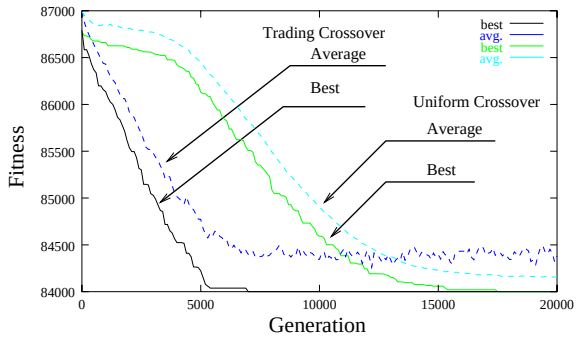

図 15: 代表的な収束曲線 (H-sum-sum)

6.5.3 考察

収束曲線を見ると、トレーディング交叉は、探索の初期の段階から直線的に最適値に向かって収束していく。これは、トレーディング交叉が良好な評価を持つ塩基配列同士の結合を保存するように働いているからだと考えられる。つまり、ビルディングブロックの生成を促進するような交叉を行っているからだと考えられる。

この実験では、トレーディング交叉と一様交叉の収束値に差は見られなかった。一様交叉であっても、最初の頃の世代では、適応度の更新に停滞が見られるものの、その後はトレーディング交叉と同じ適応度に落ち着く。

このような現象が起きる原因として考えられる事としては、問題のクラスが、新しい交叉を必要としない程度の難易度であった。ということが考えられる。この場合は、すでに最適に達しているという事になる。

もう一つの可能性は、評価関数の影響で悪平等な評価に収束してしまったというものである。H-sum-sum は確かに GA が探索しやすいような適応度地形を生成するが、和を取る演算のため、どうしても、すべての塩基配列同士の H-sum が少しづつ悪い評価になったり、逆にいくつかの塩基配列に悪い評価が集中してしまう事態が起こりやすい。実験 1 では H-sum-sum を利用して最適解を見つけだすことに成功したが、常にそのような関係が成り立つわけではないのである。

7 まとめ

本研究では、いくつかの実験を行い次のような知見を得た。実験 1 では、Garzon らが定義した評価関数 H-measure による最適化を、H-sum-sum という評価を用いて定義し直すことにより、最適値に収束することが可能になった。

また、GA の交叉としてトレーディング交叉を提案した。この交叉は形質遺伝性を考慮しており、個体内のローカルな塩基配列同士の適応度を利用する。実験 2 では、解を求めるのが困難な状況であっても、トレーディング交叉は単純な一様交叉よりも良好な結果が得られることを確認した。

さらに、相補的な配列も含めた評価関数を定義し、これにトレーディング交叉を適用してその挙動を観察した。

トレーディング交叉に関しては、探索初期段階における探索性能の高さは認められるものの、更に困難な問題に対して、その性能評価を行う必要があるであろう。

また、これまでは、ベンチマーク的な最適化問題としての解法を検討してきたが、この最適化手法を活かして、DNA Computingの研究者からのの要望に応えるような配列設計を行い、その効果を実際の反応で確かめてる事も予定している。

謝辞

本研究を進めていくにあたり、山村雅幸助教授には様々な助言と適切な示唆を示していただき、心より感謝いたしております。また、小林重信教授をはじめとする先生方からは発表会において貴重なコメントをいただき、同じく心より感謝いたしております。また、東京大学の萩谷雅己教授にはDNA Computingに関する様々な資料、情報を教えていただきました。深い感謝の意を表させていただきます。そして、最後に山村研究室の皆さん。私はなかなか行動を起こさない事が多く、様々な迷惑をおかけしました。深く反省しつつ、忍耐強くつきあって下さった皆さんに心からの感謝の気持ちを表したいと思います。

参考文献

[Adleman94] L.M.Adleman: Molecular Computation of Solutions to Combinatorial Problems, *Science,Vol.266*, pp.1021-1024,1994

[Deaton96a] R.Deaton, M.Garzon, R.C. Murphy, J.a. Rose, D.R. Franseschetti, S.E. Stevens Jr. (The Molecular Computing Group): Good Encodings for DNA-based Solutions to Combinational Problems, *Second Annual Meeting on DNA based computers, workshop* ,pp.131-140,1996

[Deaton96b] R.Deaton,M.Garzon,*et al.*: Genetic Search of Reliable Encodings for DNA Based Computation, *Late-Breaking Papers at The First Conference on Genetic Programming*, pp.9-15,1996

[Garzon98] M.Garzon, R.Deaton, L.F. Nino,Ed Stevens : Encoding Genomes for DNA Computing, *Proc. of the Third Annual Genetic Programming Conf.*, pp.684-690,1998

[Satoh96] H.Satoh, M. Yamamura, S.Kobayashi: Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation, *Proceedings of IIZUKA 1996*, pp.494-497,1996

[Adleman98] Leonard M. Adleman 著, 萩谷昌己訳: DNA コンピューターで問題を解く, 日経サイエンス 1998 年 11 月号, pp.20-29,1998

[小野 97] 小野功: 形質遺伝を重視した遺伝的アルゴリズムによる最適化, 1997 年度　博士（工学）論文, pp.10,1997

[佐藤 97] 佐藤浩, 小野功, 小林重信: 遺伝的アルゴリズムにおける世代交代モデルの提案と評価, 人工知能学会誌, Vol.12, No.2, pp.734-744,1997

[萩谷 97] 萩谷昌己: 分子コンピュータの理論と構築, 1997 年度人工知能学会全国大会論文集, pp.9-17,1997

[山村 94] 山村雅幸, 小林重信: 遺伝的アルゴリズムの工学的応用, 人工知能学会誌, Vol.9,No.4, pp.7-12,1994

A　追加データ

図 16〜18 は、実験 2 の実験条件で 10 × 20 問題を解いた場合の結果を示すグラフである。

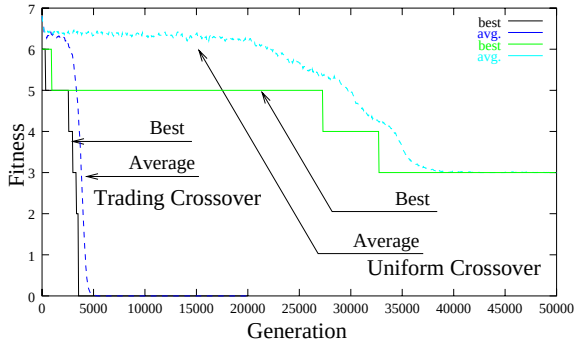

図 16: 収束曲線 (10x20 問題)

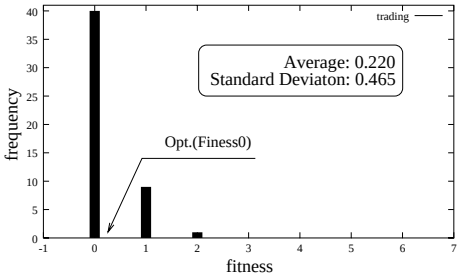

図 17: 収束解の頻度分布 (トレーディング交叉)

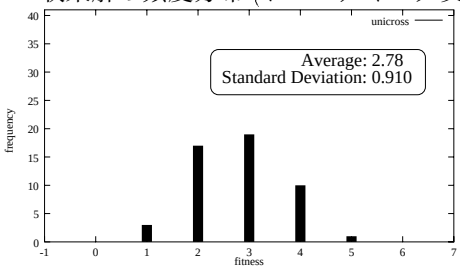

図 18: 収束解の頻度分布 (一様交叉)