平成 14 年度

修士学位論文

ウイルス感染による
最大独立集合の進化論的求解法

An Evolutionary Approach for Maximum Independent Set Problems with Virus Infection

1055107　　登 伸一

指導教員　　坂本 明雄

2003 年 1 月 31 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報システム工学コース

要 旨

ウイルス感染による
最大独立集合の進化論的求解法

登 伸一

遺伝的アルゴリズムはダーウィンの進化論に基づいた最適化手法の一種であり，基本的な操作として染色体の交叉，遺伝子の突然変異および生物集団の自然淘汰を用いる．

一方，ダーウィンの進化論以外にも多くの進化論が提案されている．ウイルス進化論はその中の 1 つであり，生物は染色体のウイルス感染によって進化するという考えに基づいている．

本研究では，最大独立集合を求めるための解法として，ウイルス進化論の考え方であるウイルス感染を用いた進化アルゴリズムを提案し，幾つかのクリーク問題から構成される DIMACS ベンチマークグラフに対して，計算機実験を行い，その有効性を示す．

キーワード　遺伝的アルゴリズム，ウイルス進化論，最大独立集合問題

Abstract

An Evolutionary Approach for Maximum Independent Set Problems with Virus Infection

NOBORI Shin-ichi

A genetic algorithm is a kind of optimization techniques based on Darwinism, and the crossover of chromosomes, the mutation of gene, and the natural selection of the group are used as basic operations.

On the other hand, the evolution theories are not only Darwinism but also lots of theories. One of these theories is the virus theory of evolution, which asserts that the evolution of living things is based on the virus infection of the chromosome.

In this report, an evolutionary approach for maximum independent set problems using virus infection is proposed. Effectiveness of the proposal technique is described by computer experiments for DIMACS benchmark graphs which are originally collected from several clique problems.

key words Genetic Algorithm, Virus Theory of Evolution, Maximum Independent Set Problems

目次

図目次

表目次

第 1 章

序論

与えられたグラフにおける最大独立集合を求める問題は，節点数の多項式時間では解くことはできないと予想されている NP-困難な問題の一つである．また，この最大独立集合問題は補グラフに対する最大クリーク問題と等価であり，実用的な時間で近似解を求める種々の方法が研究されている [1].

一方，生物の進化を模倣した最適化手法の一種として，遺伝的アルゴリズム (Genetic Algorithm: GA) がある．このアルゴリズムは自然界における生物の進化モデル，すなわち世代を形成している個体の集合 (解候補) の中で，環境への適応度の高い個体が次世代により多く生き残り，また複製，交叉および突然変異を繰り返しながら次世代を形成していく過程を模した最適化法であり，より環境に適した適応度の高い個体を得ようとするアルゴリズムである [2].

この GA を最大独立集合問題に適用した手法として文献 [3] や文献 [4] などが発表されている．特に文献 [4] では，DIMACS というベンチマークグラフに適用し，その結果をヒューリスティック手法 CBH[5] と比較している．しかし，文献 [4] の結果を見る限り，すべての DIMACS ベンチマークグラフに適用していない点，発見できている解が CBH と同程度である点など，十分な結果であるとはいえない [6].

さらに文献 [7] では，一つの染色体から複数の独立集合をデコードし，その最良解を採用するという改良を加えることで，従来手法より良い結果を得られたことが報告されている．しかし，文献 [7] の手法において，GA の基本的操作の一つである交叉に改良の余地があると思われる．

本論文では，その問題点に着目し，生物は染色体のウイルス感染によって進化するという

ウイルス進化論 [8][9] の考え方を導入して，“交叉の代わりにウイルス感染” を用いた遺伝的アルゴリズムを提案する．そして，DIMACS ベンチマークグラフに対して計算機実験を行い，文献 [7] の結果と比較し，その有効性を述べる．

2 章以降の概要を以下に示す．

2 章では，最大独立集合問題について述べる。3 章では，遺伝的アルゴリズムについて簡単に説明し，4 章では，文献 [7] の解法であるマルチデコード GA について述べる．次に 5 章では，ウイルス進化論について簡単に説明し，新しいアルゴリズムを提案する．そして，6 章で計算機実験による結果を示し，結論と今後の課題を 7 章で述べる．

第 2 章

最大独立集合問題

与えられたグラフにおける最大独立集合を求める問題は，節点数の多項式時間では解くことはできないと予想されている NP-困難な問題の一つである．また，この最大独立集合問題は補グラフに対する最大クリーク問題と等価であり，実用的な時間で近似解を求める種々の方法が研究されている [1].

本章では，はじめに最大独立集合問題を考える前段階としてグラフ理論について簡単に説明する．また，最大独立集合問題が含まれる組み合わせ最適化問題を例を挙げて説明し，最後に最大独立集合問題について述べる．

2.1 グラフ理論

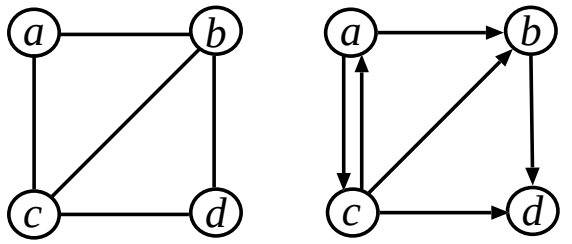
グラフ理論は 18 世紀半ばにスイスの数学者オイラー (L. Euler) により提唱された “ケーニヒスベルグ (Könisgsberg) 橋の問題” がそのはじまりといわれている．したがって，グラフ理論はそれ自身かなり長い歴史を有するものであるが，最近では組み合わせ理論あるいは組み合わせ最適化などと呼ばれる分野，あるいはオペレーションズリサーチなどの分野の発展に伴い，これらの分野の基礎として重要なものの一つとなっている．

2.1.1 グラフの定義

グラフ (graph) とは，頂点 (vertex) の有限集合 V と辺 (edge，arc) の有限集合 E とによって定義される．したがって，グラフを組 (V, E) で表記することがある．ここで，辺とは頂点のペア (二つ組)($\in V \times V$) である．頂点，辺は，それぞれ節点 (node)，枝 (branch)

と呼ばれることもある．

A，B を集合とすると，$A \times B$ は “A の要素と B の要素のあらゆる組合せの全体” を意味し，A と B の直積という．すなわち，$A \times B = \{(a,b)|a \in A, b \in B\}$．$A$ と B の 2 項関係とは，$A \times B$ の部分集合のことである．



(a) 無向グラフ G_1　　(b) 有向グラフ G_2

図 2.1　グラフ

グラフは，図 2.1 の例に示すように，図で表現することができる．図 2.1 中の (a) のグラフ G_1 と (b) のグラフ G_2 は次のようなグラフである．

$$G_1 = (V_1, E_1)$$

$$V_1 = \{a, b, c, d\}$$

$$E_1 = \{[a,b], [a,c], [b,c], [b,d], [c,d]\}$$

$$G_2 = (V_2, E_2)$$

$$V_2 = \{a, b, c, d\}$$

$$E_2 = \{(a,b), (a,c), (b,d), (c,a), (c,b), (c,d)\}$$

図 2.1 では，頂点を○で，辺を線で表している．辺は，矢印が付かない場合と付いている場合がある．前者を無向グラフ (undirected graph)，後者を有向グラフ (directed graph, digraph) という．このことを明確にするために，図 2.1 における矢印の付いた辺すなわち有向辺は，(u,v) のように丸括弧で囲んで表記する．無向グラフにおける辺 $[u,v]$ は，二つの有向辺 (u,v) と (v,u) をまとめて表現したものと考えることができる．したがって，一般に，$[u,v] = [v,u]$ であるが，$(u,v) \neq (v,u)$ である．

また, 無向グラフにおいて, ある頂点に接続している辺の個数を, その頂点の次数 (degree) という．次数 0 の頂点は孤立 (isolated) しているという．どの 2 頂点をとっても，それらが隣接しているようなグラフを完全グラフ (complete graph) という．頂点の個数 $|V|$ が n の完全グラフを K_n と表す．有向グラフの場合は，ある頂点について，それを始点とする辺の個数をその頂点の出次数 (out-degree)，逆に終点とする辺の個数を入次数 (in-degree) という．

2.2　組み合わせ最適化問題

最適化問題の中でも，解が整数などの集合，あるいは，順列で与えられている場合の問題のことを組み合わせ最適化問題と呼ぶ．システムの計画や運用などの効率化を考える場合，多くの問題が組み合わせ最適化問題として定式化できるが，実際的な問題の多くは厳密な最適解を求めるのは困難である．

与えられたグラフにおける最大独立集合を求める問題は，組み合わせ最適化問題であり，個別問題のサイズの多項式時間では解くことはできないと予想されている NP-困難な問題の一つである．

NP とは，非決定性アルゴリズムによって，個別問題のサイズの多項式のオーダーの時間計算量で解くことのできる決定問題のクラスである．また，問題 Q に対し，NP に属するすべての問題をそれぞれ Q に帰着できるとき，Q は NP-困難であると呼び，特に NP に属する NP-困難な問題は，NP-完全であると呼ぶ.

NP-完全である問題 Q を解く効率の良いアルゴリズムを発見することができれば，そのアルゴリズムを用いることによって，他のすべての NP-完全問題や NP に属するすべての問題を解く効率の良いアルゴリズムが自動的に構成できることが知られている.

したがって，最大独立集合問題を解く効率の良いアルゴリズムを発見することができれば，そのアルゴリズムを用いることによって，他の NP-困難な組み合わせ最適化問題も同様に効率良く解くことができる．以下に，NP-困難な組み合わせ最適化問題の例を示す.

2.2.1　巡回セールスマン問題

いくつかの都市があり，あるセールスマンが各都市を一度すつ巡回訪問しなければならないものとする．この際，各都市間の距離は決まっているものとし，巡回路長 (総距離) が最小となるように巡回路を決定する問題を，巡回セールスマン問題 (Traveling Salesman Problem，TSP) と呼ぶ.

いま，都市数を N とし，都市には 1 から N までの番号が付けられているものとする．また，都市 i，j 間の距離を d_{ij} とする．決定変数として都市の訪問順序を $(1,\ldots,N)$ の順列 $(t_1,\ldots,t_N)$ で表すと，この問題は

$$min\{t_i\}\sum_{i=1}^{N} d_{t_i t_{i+1}}$$

と表される．ただし，$t_{N+1}=t_1$ である．

2.2.2　ナップザック問題

いくつかの荷物と一定重量までの荷物を入れられる袋がある．各荷物の重量およびその価値 (重要度) は分かっているものとする．このとき，袋の許容重量以内でその価値の和が最大となるように入れる荷物を決定する問題を，ナップサック問題 (Knapsack Problem, KP) と呼ぶ.

図 2.2 の例は，荷物の重量が数字，価値は各荷物の大きさ，ナップザックの重量制限は 20kg 以内となっていることを示している.

いま，荷物が全部で N 個あるとして，順に 1 から N までの番号が付けられているものとし，荷物 i の重量および価値をそれぞれ a_i および c_i，また，袋の許容重量を b とする．このとき，決定変数 x_i を導入し，荷物 i を袋に入れることを $x_i=1$，入れないことを $x_i=0$ で表すと，この問題は以下のように定式化できる.

$$max\{x_i\}\sum_{i=1}^{N} c_i x_i$$

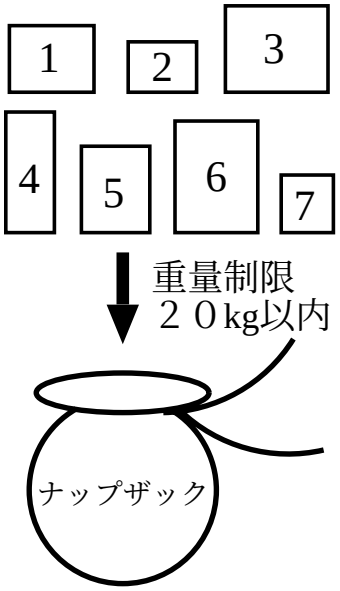


図 2.2　ナップザック問題の例

$$subject\ to \sum_{i=1}^{N} a_i x_i \leq b$$
$$x_i \in \{0,1\} (i = 1, \ldots, N)$$

また，ナップザック問題は，以下に示すような多くの応用例の基礎となっている．

1. 一定予算内のでの物資の購入
2. 従業員の能力に基づいた人事管理
3. 金融取り引き意志決定 (一定の資金で，リスクが最小になるように株，債券などへの投資法を決める)
4. 貨物輸送システム

2.3　最大独立集合問題

与えられたグラフ $G = (V, E)$ の節点集合 V の部分集合 U において，U に属するどの 2 節点間にも枝が存在しないとき U を G の独立集合という．U が独立集合であり，U を真に含む G の独立集合が存在しないとき U を極大独立集合，最大個の節点をもつ極大独立集

合を最大独立集合という．

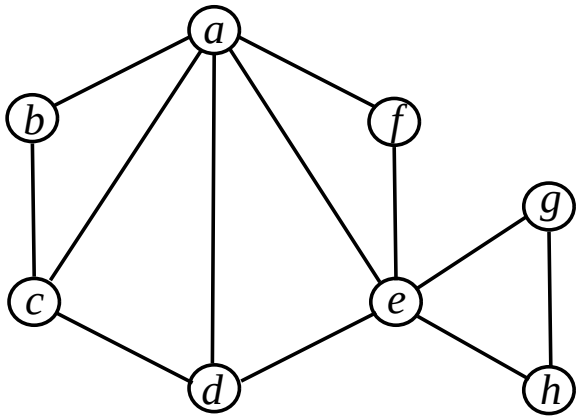


図 2.3　グラフ G

最大独立集合の求める例として，図 2.3 のようなグラフ G を考える．このとき，極大独立集合は，$\{c, f, g\}$ や $\{c, h, g\}$，$\{b, d, f, g\}$ が挙げられる．また，このグラフには 5 個の節点をもつ独立集合は存在しないことから，最大独立集合は $\{b, d, f, g\}$ であり，その大きさは 4 となる．

このように，与えられたグラフにおいて最大個の節点をもつ独立集合を求める問題が最大独立集合問題である．

2.4　まとめ

本章では，最初に最大独立集合問題を考える前段階としてグラフ理論について説明した．そして，NP-困難な組み合わせ最適化問題の例を挙げて説明し，最後に，最大独立集合問題について述べた．次章では，遺伝的アルゴリズムについて説明する．

第 3 章

遺伝的アルゴリズム

遺伝的アルゴリズム (Genetic Algorithm: GA)[2] は，生物進化の原理に着想を得たアルゴリズムで，最適化手法の一つである．すなわち，生物の進化と同様にその世代を形成している個体の集合 (解候補) を用い，その中で環境への適応度の高い個体を次世代に多く残しながら，交叉や突然変異によって新しい個体を生成して次世代を形成する．それにより，より環境に適した適応度の高い個体が生み出されていく．遺伝的アルゴリズムはこのような過程を繰り返すことによって最適な解を得ようとする探索アルゴリズムであり，大域的かつ並列的な探索手法である．

本章では，遺伝的アルゴリズムの流れについて説明する．

3.1 概要

一般的に生物は環境にうまく適応できると，生命の存続と繁殖が可能となる．繁殖を行う際には，その個体が持っている生物としての設計情報を何らかの形で子孫に伝授しなければならない．そのような設計情報は染色体の特定の位置に存在する遺伝子に書き込まれている．

遺伝子によって生物としての情報伝達が親から子へ行われるが，高等生物の繁殖は有性生殖であり，この場合は父方と母方の遺伝子が混ざり合ったものを遺伝子として受け継ぐことになる．この過程は交叉と呼ばれている．一方，遺伝子のコピーを行う際などに微妙なエラーが生じることがあり，これは突然変異と呼ばれる．次の世代には，各個体の中でもより優れた，つまり環境への適応度の高い個体の遺伝子情報が優先的に伝えられる．適応度の低

い個体は短命であったり，増殖できなかったりするからである．同時に適応度の低い個体は自然淘汰されていく．このような原理に基づいて世代を重ねていくと，次第に環境への適応度が高い個体が多くなっていく．これが遺伝と進化の基本的な原理である．

自然界での個体を特徴づけているものは，遺伝子が一定の順序で配列している染色体であり，遺伝的アルゴリズムでは有限固定長の記号列を染色体に，個々の記号を遺伝子に対応させている．

m 個の個体から成る世代 t の生物集団を $M(t)$ としたとき，$M(t)$ は同一の形質をもつ複数の個体が存在するため多重集合になる．自然界の生物は複数の染色体をもつが，遺伝的アルゴリズムでは個体 i は 1 つの染色体 α_i をもつものとする．異なる遺伝子の要素から成る集合を Z としたとき，α_i $(i = 1, 2, \ldots, m)$ は，Z の要素 Z_{ij} の並び $\alpha_i = (Z_{i1}, Z_{i2}, \ldots, Z_{in})$ によって作られている．このとき，n は染色体の長さである．

また，染色体の集合を遺伝子空間と呼び，染色体上で遺伝子の置かれる位置を遺伝子座，そこに配置され得る遺伝子を対立遺伝子，遺伝子の配列からなる染色体を遺伝子型，その遺伝子の並びによって定まる発現を表現型と呼ぶ．ただし，最適化問題を解く場合には，遺伝子型と表現型は 1 対 1 に対応しているものとする．

遺伝的アルゴリズムにおいて，生物 (個体) がどれだけ環境に適応しているのかの度合は適応度と呼ばれ，自然淘汰に対する個体の有利さを表す尺度になっている．これは個体が次の世代に残す子孫の数に対応しており，このような適応度を個体に対応させる関数は，適応関数と呼ばれている．

遺伝的アルゴリズムは，実際には図 3.1 に示すような流れで処理が行われる．以下ではこれらの過程について説明する．

ステップ 1：遺伝子型の決定 (コーディング)

遺伝的アルゴリズムでの遺伝子の要素は，DNA ではなく記号列である．したがって，まず対象とする問題を遺伝子の列の形で表現しなければならない．遺伝子のどこにどの数値，文字を割り振っていくかを決定する．つまり，“対象とする問題” から “記号列”

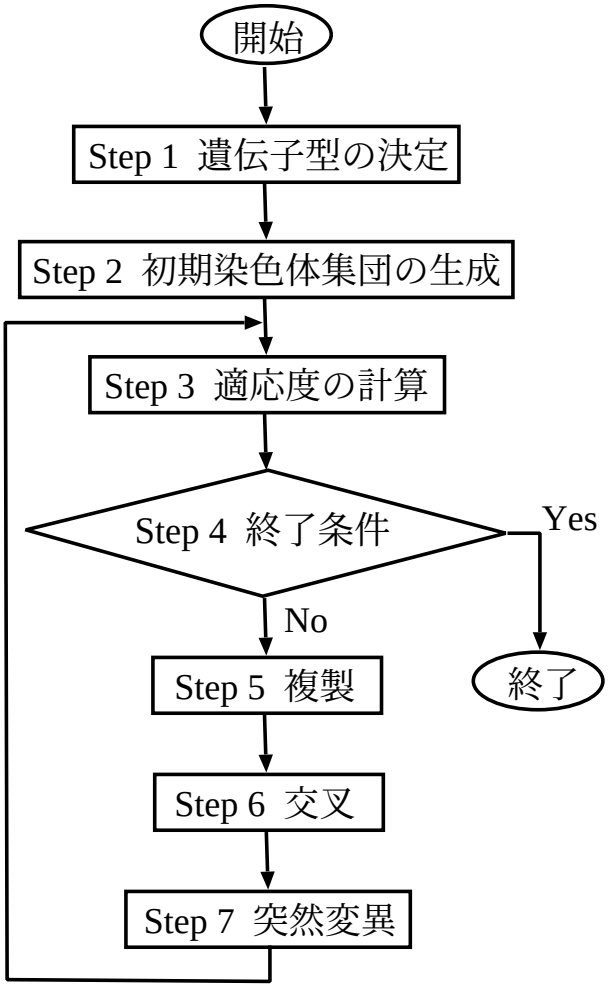


図 3.1　　GA の流れ

へという変換を行う．この一般的方法はまだ確立されておらず，設計者の腕や経験にかかっている．

ステップ 2：初期染色体集団の生成

ステップ 1 で決定した遺伝子型で，要素が異なるさまざまな個体を発生させる．個体数は，問題の難易度や性質によるが，一般的には数十以上発生させる．個体数が少ない場合，並列的な処理を特徴とする遺伝的アルゴリズムの良さが発揮されない．しかし，個体数が多い場合，一世代あたりの計算量も大きくなり，無駄が多くなる．これもまた，設計者の腕や経験にかかっている．

ステップ 3：適応度の計算

遺伝的アルゴリズムでは，“適応度” が非常に重要な要素になる．ここでは，各個体の適応度をあらかじめ定めた方法で計算する．

ステップ 4：終了条件

集団が，あらかじめ定めておいた終了条件を満足するかどうかをチェックし，満足していれば終了し，していなければ次のステップに進む．

ステップ 5：複製

ステップ 3 で求めた適応度に基づいて，次のステップで交叉を行う生存分布を決定する．

ステップ 6：交叉

２つの染色体間で遺伝子を組み換えて，新しい個体を発生させる．

ステップ 7：突然変異

遺伝子のある部分の値を強制的に変えて，集団としての多様性，つまりばらつきを大きくする．これにより，より良い解をもつ個体の発生を期待する．しかし，この突然変異の割合をあまり大きくしすぎると悪い方向への変異の確率も大きくなり，良い解がなかなか求まらなくなる．

ステップ 8：繰り返し

ステップ 3 に戻り，各個体の適応度を評価する．

3.2　複製

複製は，集団の中での適応度の分布にしたがって，次のステップで交叉を行う個体の生存分布を決定する操作である．ここでは，基本的な複製方法である適応度比例戦略を用いて説明する．

3.2.1　適応度比例戦略

適応度比例戦略は，各個体の子孫はその適応度に比例した確率で選ばれる．したがって，適応度の大きい個体ほど選択されやすく，次のステップの交叉に参加できる可能性が高くなる．

実際には，親の個体群から次世代の子の個体群を生成するとき，まず子の個体群を空にし， 親の個体群から各個体の適応度に比例した確率，つまり全個体の適応度の総和に対する各個体の適応度の比率で個体を選び，子の個体に加える．子の個体として選ばれた親の個体は除去せず， 親の個体群に含まれたままとする． 図 3.2 に適応度比例戦略における子の個体の選び方の例を示す．図 3.2 の例では 5 個体の親から子を選ぶが，その確率は適応度によって定まる．例では個体 A が確率 0.4 で最も大きい．一つ目の子の個体として個体 A が選ばれたとしても，二つ目の子の個体を選ぶときは同じ確率で個体を選ぶ．これは同じ個体が複数回選ばれる可能性があることを示す．これを繰り返し，子の個体数が設定数に達したら複製の終了となる．

また，適応度比例戦略では適応度の高い個体が複数回選ばれる可能性が高くなり，逆に適応度の低い個体はほとんど選ばれないことになる．このことから，適応度比例戦略は探索を重点化する効果がある．しかし，探索が進み，個体の適応度にあまり差がつかなくなると，良い個体が選択され難くなる．また，適応度の差が極端に大きいと，高い適応度を持つ個体のみが急速に増殖し，多様性を失い，かえって探索効率を低下させてしまう．

3.3　交叉

交叉は，2 つの染色体間で，遺伝子を組み換えて，新しい個体を発生する操作である．ここでは，各交叉手法について説明する．

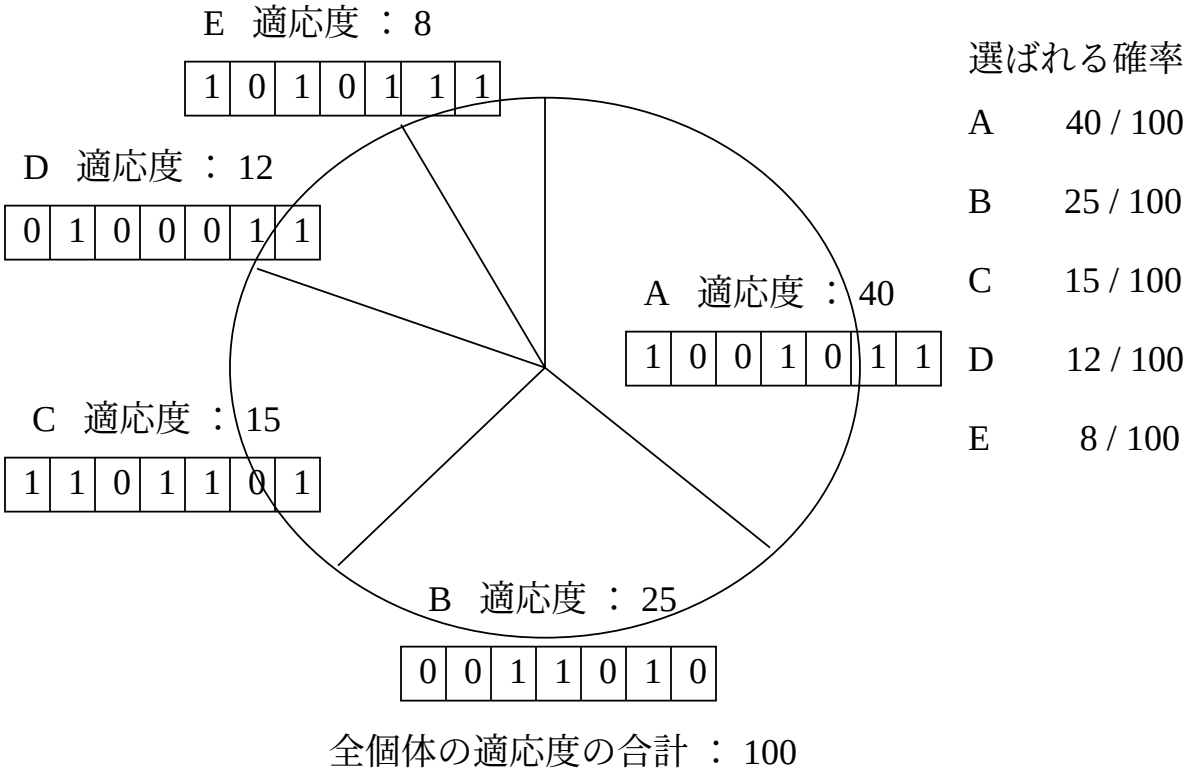


図 3.2 適応度比例戦略

3.3.1 Order crossover (OX，順序交叉)

OX は，親染色体の交叉箇所をもう一方の親染色体の遺伝子順序に組み換え，子染色体を生成する交叉である．図 3.3 に，交叉手法 1-point-right OX の場合の親，子染色体を示す．図中，“∗” はランダムに選ばれた交叉点である．また，1-point はランダムに選ばれた交叉点 “∗” の右側または左側を，2-point はランダムに選ばれた 2 つの交叉点 “∗” と “∗” の内側または外側を交叉させることを意味している．

Parent 1 : e d a ∗ f c h b g　　Offspring 1 : e d a ∗ b c f g h
Parent 2 : a b c ∗ d e f g h　　Offspring 2 : a b c ∗ e d f h g

図 3.3 1-point-right OX

図 3.3 に示すように，1-point-right OX により，子 1 の交叉点より (交叉点も含まれ

る) 前半部は親 1 の遺伝子をそのまま継承し，後半部は親 2 の遺伝子順序に組み換えられる．子 2 に対しても同様の操作を行う．図 3.4，3.5，3.6 に，それぞれ 1-point-left OX，2-point-inside OX，2-point-outside OX の親，子染色体を示す.

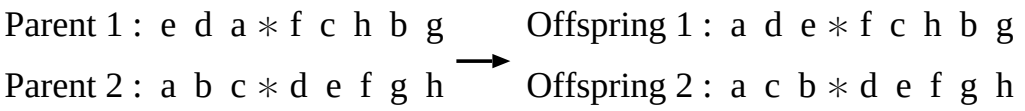


図 3.4 1-point-left OX

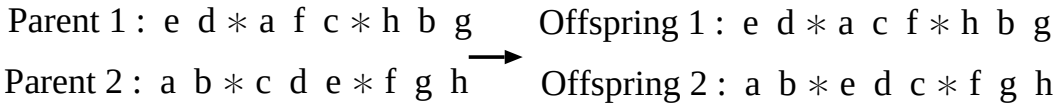


図 3.5 2-point-inside OX

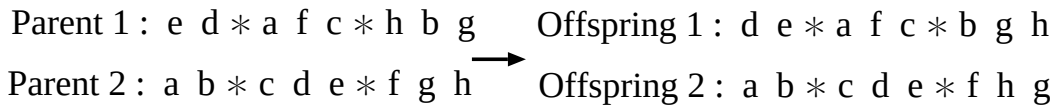


図 3.6 2-point-outside OX

3.3.2 Partially mapped crossover (PMX，部分写像交叉)

PMX は，親染色体の交叉箇所を子染色体に継承し，致死遺伝子が発生しないように部分的にパッチをあてる交叉である．図 3.7 に，交叉手法 1-point-right PMX の場合の親，子染色体を示す.

図 3.7 に示すように，1-point-right PMX により，子 1 の交叉点より後半部は親 2 の交叉箇所をそのまま継承する．前半部は親 1 の遺伝子を継承するが，その後，子 1 に致死遺伝子が発生しないように親 1 の親 2 に対する写像をあてる．例えば図 3.7 の場合，遺伝子 a はそのまま親 1 から継承しても致死遺伝子とはならない．しかし，遺伝子 d は致死遺伝子となるので，d の親 2 に対する写像である遺伝子 b を継承する．同様に，遺伝子 e に対し親 2 に対

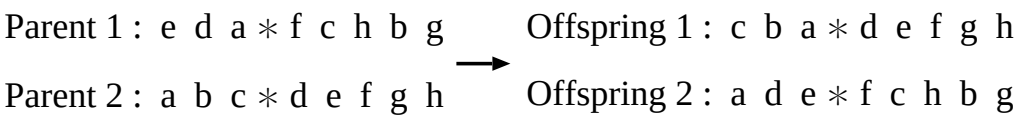


図 3.7　1-point-right PMX

する写像 a を継承するが，これも致死遺伝子となるのでさらに写像をたどり，a に対する写像遺伝子 c を継承する．子 2 に対しても同様の操作を行う．図 3.8，3.9，3.10 に，それぞれ 1-point-left PMX，2-point-inside PMX，2-point-outside PMX の親，子染色体を示す．

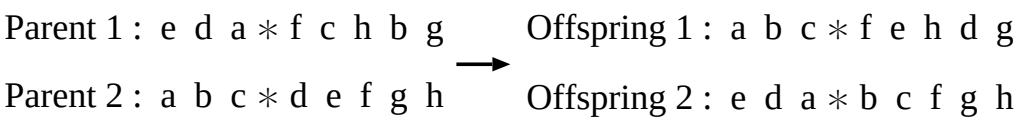


図 3.8　1-point-left PMX

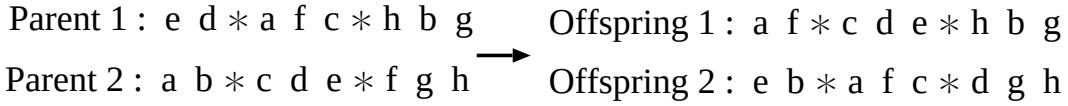


図 3.9　2-point-inside PMX

3.3.3　Cycle crossover (CX，周期交叉)

CX は，両親染色体の遺伝子の cycle を求め，その遺伝子を子染色体の各遺伝子座に継承する．図 3.11 に示すように，親 1 の遺伝子座 1 の遺伝子を cycle の開始点として親 2 への写像をたどり，e → a → c → e という cycle を得る．これが子 1 の各遺伝子座に継承される．次に，親 2 の遺伝子座 2 の遺伝子を cycle の開始点として，b → d → f → h → g → b の cycle を得，子 1 の各遺伝子座に継承する．この操作を cycle が得られなくなるまで繰り返し，子 2 に対しても同様の操作を行う．

CX により，子染色体は両親染色体の遺伝子をそれぞれ 0.5 の確率で継承する．しかし，子染色体が両親染色体と同じ染色体になる可能性もある．

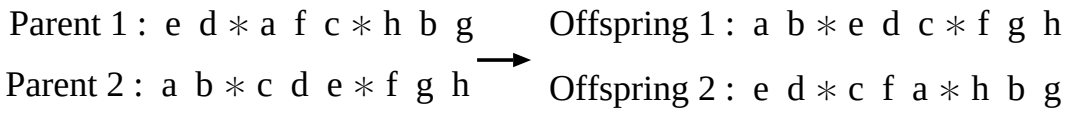


図 3.10　2-point-outside PMX

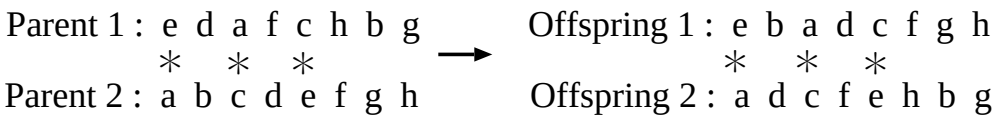


図 3.11　CX

3.4　突然変異

突然変異は，遺伝子のある部分の値を強制的に変える操作である．

ここでは，最も単純な突然変異を用いて説明する．図 3.12 に示すように，染色体中の遺伝子をランダムに 2 点選び交換する．図 3.12 では，ランダムに選ばれた 2 点 c と g が交換されることになる．

図 3.12　単純突然変異

3.5　特徴

遺伝的アルゴリズムは，単独の個体ではなく生物集団を用いること，および個体の適応度（適応関数の値）自体を用いることに特徴がある．状態空間における単独の点ではなく，状態空間の広い範囲にわたって散在する点の集合について，適応度を求め，最適値に近い値を与える点の集合を再構成し，交叉や突然変異によって次の点の集合を作るという操作を繰り返す．よって，遺伝的アルゴリズムは，最適化問題における局所解からの脱出という課題を点の集合を用いることによって解決しているのである．つまり，遺伝的アルゴリズムは，局

所解に陥る可能性が低いアルゴリズムといえる．

3.6　まとめ

本章では，遺伝的アルゴリズムの原理について簡単に説明し，アルゴリズムの流れと共に，遺伝的アルゴリズムの基本的操作を説明した．次章では，最大独立集合問題を解くために，遺伝的アルゴリズムにマルチデコードを適用した解法とその問題点について述べる．

第 4 章

マルチデコード GA

最大独立集合問題に対して今まで種々の解法が提案されている [1]．中でも，マルチデコード GA[7] による解法が有効であることが報告されている．しかし，その解法において GA の基本的操作の一つである交叉に改良の余地があると見受けられる．

本章では，文献 [7] で提案されているマルチデコード GA の解法を説明し，その解法の問題点を述べる。

4.1 マルチデコード GA

先に提案されているマルチデコード GA[7] による解法の流れ (図 4.1) を説明する．

1. 乱数を用いて初期染色体集団を生成する．
2. 各染色体に対してマルチデコードを適用し，得られた独立集合のサイズを用いて，適応度を求める．
3. 適応度に応じた確率で染色体を複製する (適応度比例戦略)．
4. 交叉確率 p_c で，交叉を行う．
5. 突然変異確率 p_m で，突然変異を行う．
6. 終了条件を満たすまで 2〜5 を繰り返す．
7. 得られた集団中で最も適応度の高い個体の染色体から最大独立集合を出力する．

以下では，各々の働きについて説明する。ここで，GA の基本的操作である複製，交叉，突然変異においては，3 章で説明した方法を用いる．

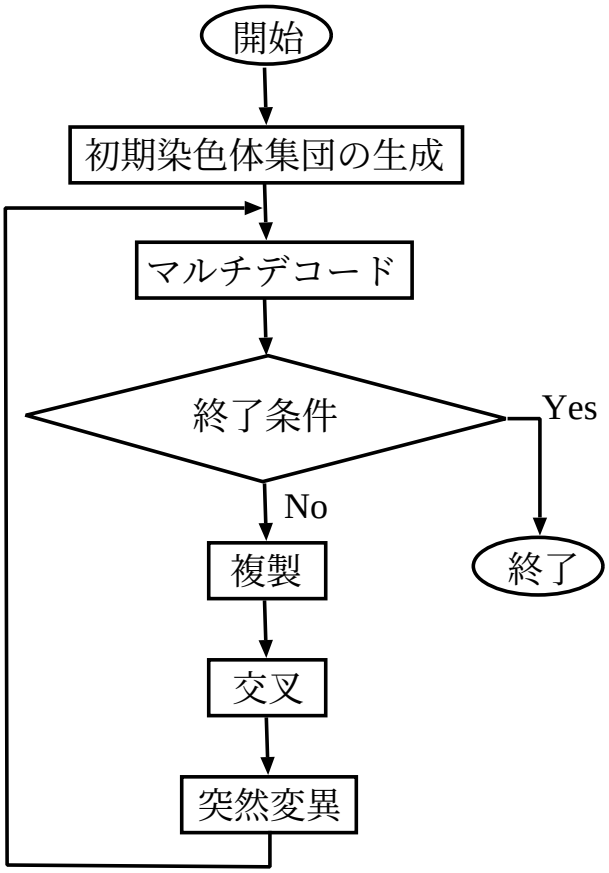


図 4.1　マルチデコード GA の流れ

4.2　マルチデコード

図 4.2 に示すように，染色体 α は “グラフの全節点の順列” である．

図 4.2　染色体 α

染色体 α から極大独立集合 A を抽出する次の操作をデコードと呼ぶ．デコードは，まず α の先頭節点 a を A の要素とすることから始める．以後は，α に並んでいる順に節点 x を調べ，x を A に含めても A が独立集合であれば x を A に含め，そうでなければ x を A に含めないという操作を最後の節点まで続ける．この操作で得られた節点集合 A はグラフ

の極大独立集合であり，α の適応度は A に含まれる節点数とする．

マルチデコードは，上記のデコードで得られた極大独立集合 A を構成する節点を染色体 α の先頭から順に並び換えた後，さらに A に含まれなかった残りの節点について同様のデコード操作を繰り返し，あらかじめ指定された上限値 k_{max} 個の極大独立集合を求めるまで続ける．したがって，マルチデコード操作が終了した α は次のような構成になっている．

α = (独立集合$_1$ を構成する節点集合,
独立集合$_2$ を構成する節点集合,
...
独立集合$_k$ を構成する節点集合,
どの独立集合にも属さない節点)

ただし，$k \leqq k_{max}$ であり，独立集合$_1$ は k 個の独立集合の中でサイズが最大のものである．

このとき，独立集合$_1$ のサイズを用いて適応度を求める．

4.3　終了条件

終了条件は，世代数 T_{stop} をあらかじめ与えておき，T_{stop} 世代にわたって，最良解が一度も改善されないとき，収束したと見なして終了する．

4.4　マルチデコード GA の問題点

ここでは，マルチデコード GA で最終的に用いられている 2-point-inside PMX を例に挙げて，マルチデコード GA における交叉手法の問題点について述べる．

2-point-inside は交叉範囲を示しており，2 つの交叉点をランダムに決め，その内側を対象に交叉を施している．partially mapped crossover とは，3.3.2 で説明したように親染色体の交叉箇所を子染色体に継承し，致死遺伝子が発生しないように部分的にパッチをあてる交叉である．

4.2 で述べたように，マルチデコード GA における染色体は “グラフの全節点の順列” で

あり，遺伝子はマルチデコードにより，複数個の独立集合を順に並べた形で配置されている．例えば，以下の染色体 α のように，独立集合 A を構成する節点 a_1〜a_9，続いて独立集合 B を構成する節点 b_1 ...と並べられているとする．この

$$\alpha = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, b_1, b_2, b_3, \ldots)$$

に交叉を施した場合を考える．交叉範囲として a_5〜a_7 が選ばれ，もう一方の親から f_5〜f_7 が交叉されたとする．これを

$$\alpha' = (a_1, a_2, a_3, a_4, f_5, f_6, f_7, a_8, a_9, b_1, b_2, b_3, \ldots)$$

とすると，α' のデコードは，まず α' の先頭遺伝子 a_1 から順に独立集合が形成できるか判断していく．したがって，a_1〜a_4 は独立集合を形成することになる．次に f_5 は独立集合を形成するかどうかを判断するわけであるが，a_1〜a_4 の独立集合が既に形成されていることから，f_5 が含まれず，染色体の後方の a_5〜a_7 が含められ，元の独立集合 A が形成される可能性が比較的高い．仮に f_5 が独立集合を形成したとしても，続く f_6，f_7 が形成される保証はなく，a_1〜a_4 との枝関係から形成されないことも考えられる．このように，交叉により f_5〜f_7 が染色体に組み入れられても，その影響で独立集合 A が改変される可能性は低いと考えられる．

次に，交叉範囲に a_1〜a_3 が選ばれ，もう一方の親から f_1〜f_3 が交叉された場合を考える．これを

$$\alpha'' = (f_1, f_2, f_3, a_4, a_5, a_6, a_7, a_8, a_9, b_1, b_2, b_3, \ldots)$$

とすると，この場合も先と同様に，α'' の先頭遺伝子 f_1 から順に独立集合が形成できるか判断していくことになる．したがって，f_1〜f_3 は独立集合を形成することになる．次に a_4 は独立集合を形成するかどうか判断するわけであるが，f_1〜f_3 を含む独立集合が既に形成されていることから，a_4 が含まれず，染色体の後方に置かれた f_4 ...が含められ，もう一方の親の元の独立集合 F が形成される可能性が比較的高い．すなわち，新たな独立集合が形成される可能性は，もう一方の親の f_4 が突然変異で a_4 と交換された場合と大差ないと考えられる．

本来，交叉という遺伝的操作は，もう一方の親が持つ “良い特徴” を受け継ぎ，それを自分の遺伝子に組み入れて，より良い子供を作ろうという目的の操作である．しかし，先に述べたマルチデコード GA のコーディング・デコーディング方法の場合，そのような可能性は比較的少ないと思われる．

4.5　まとめ

本章では，以前に提案されているマルチデコード GA の解法を簡単に説明した．また，マルチデコード GA における交叉手法の問題点について述べた．次章では，その問題点の対応策として，ウイルス進化論の考え方を導入し，“交叉の代わりにウイルス感染” を用いた遺伝的アルゴリズムを提案する．

第 5 章

ウイルス進化論を用いた GA

ウイルス進化論は，生物はウイルスの感染によって進化するという考えに基づいている．

また，ウイルスは生命体ではなく，個体から個体へ遺伝子を運ぶ生物器官の 1 つと考えられている．これは，ウイルスが染色体の一部遺伝子を置き換える機能を持つ生物の一部であり，それ自体が単独で集団を形成したり，進化することはないことを意味する [8][9].

本章では，一般的なウイルス進化論について簡単に説明し，最大独立集合問題に対して，ウイルス進化論の考え方を導入し，“交叉の代わりにウイルス感染” を用いた遺伝的アルゴリズムを提案する．

5.1　ウイルス進化論

ウイルス進化論は，ある個体の遺伝子が他の個体へとウイルスを介して移動される可能性があるという点に着目している．つまり，その個体の子孫でなくとも，ある個体の性質を引き継ぐことがある．従来からの親子間の垂直遺伝の他に，ウイルスによる個体から個体への水平遺伝があり得るということである．

次に，細菌，植物，昆虫，脊椎動物，すべての生物は，それぞれを宿主とするウイルスを持っており，ウイルスの宿主とならないただ一つの存在は，ウイルス自身であり，ウイルスはウイルスに感染されない．

また，ウイルス進化論が主張している水平 (感染) 遺伝は，ある新しい遺伝情報 (変異) は，感染域の同じ個体すべてに，一斉に伝達することができる．

そして，今までの考え方では，新しい遺伝情報を他の個体に伝えることができるのは，交

配・出産 (垂直遺伝) によってのみである．確かに，垂直遺伝によっても徐々に変異は伝わる．ただし，これだけでは環境の変動に対応して変異が起こり，それが種を変化させるほどのスピードにはなり得ない．それに，優れた形質のものがその種の大勢を占めるようになることは確かであるが，必ずしも劣った形質のものが消滅してしまうわけではなく，一定の割合で存続していくというような傾向もある．けれども，水平 (感染) 遺伝の概念では，ある種が別な種へ変化するという現象を自然に説明できる．

5.2　提案するアルゴリズムの流れ

本研究で提案するアルゴリズムの流れ (図 5.1) を説明する．

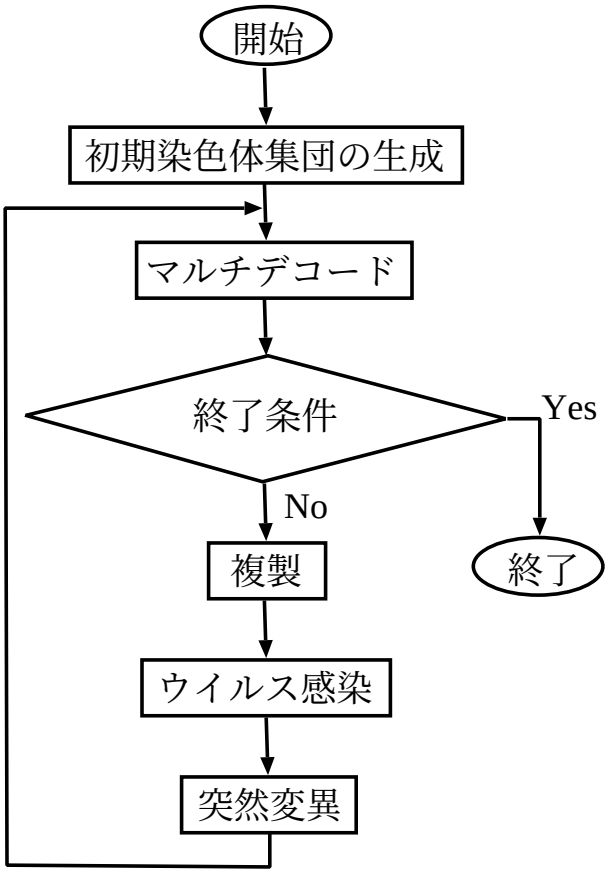


図 5.1　提案するアルゴリズムの流れ

1. 乱数を用いて初期染色体集団を生成する．

2. 各染色体に対してマルチデコードを適用し，得られた独立集合のサイズを用いて，適応度を求める．
3. 適応度に応じた確率で染色体を複製する (適応度比例戦略)．
4. 集団中の最も適応度が高い染色体からウイルスを決定し，感染確率 p_v で，ウイルスを感染させる．
5. 突然変異確率 p_m で，突然変異を行う．
6. 終了条件を満たすまで 2〜5 を繰り返す．
7. 得られた集団中で最も適応度の高い個体の染色体から最大独立集合を出力する．

5.3　提案手法

本研究で提案するアルゴリズムは，GA の基本的操作の一つである “交叉の代わりにウイルス感染” を用いる．

ウイルス感染は以下の方法で行う．

1. 各世代の最良解を得た個体の先頭から 3 番目までをウイルスとして決定する．
2. 各個体の先頭の遺伝子座に感染確率 p_v により，ウイルスの中から一つを書き込む (感染させる)．

次に具体例を示す．

各世代の最良解を得た個体の染色体を

$$\alpha = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, b_1, b_2, b_3, \ldots)$$

とする．まず，先頭から 3 番目までの遺伝子 (a_1, a_2, a_3) をウイルスとして決定する．

次に，各個体の先頭の遺伝子座に，感染確率 p_v により，a_1，a_2，a_3 の中から一つを書き込む (感染確率は,，a_1 が $p_v/2$，a_2 が $p_v/4$，a_3 が $p_v/4$ とする)．感染された染色体においては，例えば,

$$\beta = (a_1, f_2, f_3, f_4, f_5, f_6, f_7, g_1, g_2, g_3, g_4, f_1, \ldots)$$

のように，a_1 と f_1 を入れ換える．

5.4　まとめ

本章では，ウイルス進化論について簡単に説明し，そして，ウイルス進化論の考え方を導入し，GA の基本的操作の一つである “交叉の代わりにウイルス感染” を用いた遺伝的アルゴリズムを提案した．次章では，DIMACS ベンチマークグラフに対して，提案したアルゴリズムを用いた計算機実験による結果を示す．

第 6 章

実験

5 章で提案した “交叉の代わりにウイルス感染” を用いた遺伝的アルゴリズムを用い，最大独立集合問題に対して，計算機実験を行った．ここで，最大独立集合問題のベンチマークグラフとして，DIMACS ベンチマークグラフを用いた．

次に，実験で使用するパラメータを表 6.1 に示す．

表 6.1　パラメータ

個体数	M
終了世代数	T_{stop}
デコード数	k_{max}
ウイルス感染確率	p_v
突然変異確率	p_m

表 6.1 の個体数 M は集団を形成する個体の数を，終了世代数 T_{stop} は終了条件で T_{stop} 世代にわたって最良解が一度も改善されないとき収束したと見なして終了する世代数を，デコード数 k_{max} はマルチデコードの操作で形成する独立集合の上限値を表している．また，ウイルス感染確率 p_v，突然変異確率 p_m はそれぞれウイルス感染，突然変異の生起確率を表している．

以下に，Intel 製 PentiumII(433MHz) による計算機実験の結果を示す．

6.1　実験 1

DIMACS ベンチマークグラフの keller5 というデータに対し，ウイルス感染確率はどの程度が適しているのかを決定するための予備実験を行った．keller5 は，節点数 776，枝数 225990，最大独立集合の節点数 27 というグラフであり，文献 [7] ではサイズや難しさが DIMACS 全グラフの中で中庸であるという理由から，アルゴリズムの各種パラメータを決定するのに用いられたグラフである．実験において，ウイルス感染確率以外のパラメータは，文献 [7] で最終的に定められた値とし，ウイルス感染確率 0.1〜0.9 に対して，乱数列を変えた 10 回の試行を行った．よって，本手法のアルゴリズムのパラメータは，

1. $M = 50$
2. $T_{stop} = 50$
3. $k_{max} = 10$
4. $p_v = 0.1$〜0.9
5. $p_m = 0.9$

である．

表 6.2　DIMACS ベンチマークグラフ (keller5) に対する実験結果

Nodes	Edges	Opt.	p_v	Best	Ave.
776	225990	27	0.1	27	25.8
			0.2	27	26.2
			0.3	27	25.9
			0.4	27	26.2
			0.5	27	・26.3
			0.6	27	25.9
			0.7	27	25.9
			0.8	27	25.9
			0.9	27	25.9

表 6.2 は，グラフの節点数と枝数と最適解を Nodes と Edges と Opt. 欄にそれぞれ示し，また，ウイルス感染確率を p_v 欄に，乱数列を変えて 10 回試行したときの最良解を Best 欄に，得られた 10 個の解の平均値を Avg. 欄に示している．

6.1.1　考察

keller5 のグラフに対し，ウイルス感染確率を 0.1〜0.9 まで変えて試行した結果，すべてのウイルス感染確率で最適解 27 が求められた．しかし，10 個の解の平均値は違っており，ウイルス感染確率 0.5 の場合に，最良値 26.3 が求められた．この結果より，ウイルス感染確率は，0.5 と定めることにした．

6.2　実験 2

実験 1 で得られたウイルス感染確率 $p_v = 0.5$ を用いて，DIMACS ベンチマークグラフ全 9 シリーズ 66 種類に対して，計算機実験を行った．

実験結果を各シリーズ毎に，表にした．表中には，Data 欄に，付録 A，B のグラフに対応させた番号を記載している．付録 A，B には，グラフ名とその対応番号を Data 欄に，グラフの節点数と枝数を Nodes と Edges 欄に，グラフの密度 (完全グラフの枝数に対するグラフの枝数の割合)[%] を Density 欄に記載している．ただし，これらすべてのグラフは最大クリーク問題として用意されているものであるため，補グラフを作成してから本手法を適用している．また，Opt. 欄は，既知の最大クリークのサイズ，すなわち最大独立集合のサイズであるが，≧ 記号を付している数値は CBH[5] でその下限値として示されている値である．

これらのグラフに対して，CBH[5] で得られた解のサイズを CBH 欄に示す．なお DNR は，容量オーバーで実行されなかったことを意味している．また，マルチデコード GA[7] の結果を GA 欄に，本手法を New 欄に示す．ただし，マルチデコード GA 及び本手法の実験結果は，乱数列を変えて 10 回試行したときの最良解を Best 欄に，得られた 10 個の

解の平均値を Avg. 欄に，またその平均処理時間 [秒] を Time 欄に記載している．ただし，Time が 0.0 となっている場合は，平均処理時間が 0.1 秒未満であることを意味している．Best 欄においては，既知の最大解 Opt. が得られている部分は太字で，下限値に等しい解が得られている部分には ＝ 記号を，下限値より良い解が得られている部分には・記号を付した．そして，表中の右端欄には，従来手法では得られなかった Opt. 解が得られた場合は◎を，従来手法より良い解が得られた場合は○を，10 回の試行すべてにおいて Opt. 解が得られた場合は△を記載している．また，付録 C，D には，すべてのデータを記載している．ただし，付録 D の最下行欄は各々の合計値を表している．

6.2.1 c-fat シリーズ

c-fat シリーズは，故障診断問題に基づく問題である．

故障診断とは，ある未知のプラントの状態を推定するために，プラントからの出力を診断器 (diagnoser) と呼ばれるシステムが受け取り，未知である現在のシステムの状況 (故障か正常か) を推定するものである．

その問題として，故障診断問題があり，主にこの研究は離散事象システム論の分野で行われており，近年，ハイブリッドシステムの分野に応用されている．

表 6.3 DIMACS ベンチマークグラフ (c-fat) に対する実験結果

Data	Opt.	CBH	GA			New		
			Best	Avg.	Time	Best	Avg.	Time
1	12	**12**	**12**	12	3.0	**12**	12	3.0
2	24	**24**	**24**	24	4.0	**24**	24	4.0
3	58	**58**	**58**	58	3.0	**58**	58	2.0
4	14	**14**	**14**	14	18.0	**14**	14	17.0
5	26	**26**	**26**	26	32.0	**26**	26	31.0
6	64	**64**	**64**	64	50.0	**64**	64	48.3
7	≧ 126	**＝ 126**	**＝ 126**	126	33.6	**＝ 126**	126	32.0

6.2.2 johnson シリーズ

johnson シリーズは，符号理論に基づく問題である．

パラメータ n と w と d をもつ johnson グラフは，1 を w 個もつ長さ n のすべての 2 進ベクトルに対して，節点をもっている．2 つの節点の隣接する必要十分条件は，その節点が対応するビットベクトルのハミング距離が少なくとも d 離れていることである．

johnson グラフの独立集合は，1 つの符号に対して実行可能なベクトルの集合を表す．

表 6.4 DIMACS ベンチマークグラフ (johonson) に対する実験結果

Data	Opt.	CBH	GA			New			
			Best	Avg.	Time	Best	Avg.	Time	
8	4	**4**	**4**	4	0.0	**4**	4	0.0	
9	14	**14**	**14**	14	0.0	**14**	14	0.0	
10	8	**8**	**8**	8	1.0	**8**	8	0.0	
11	16	**16**	**16**	16	4.0	**16**	16	3.0	

6.2.3 keller シリーズ

keller シリーズは，ケラーの推測に基づく問題である．

表 6.5 DIMACS ベンチマークグラフ (keller) に対する実験結果

Data	Opt.	CBH	GA			New			
			Best	Avg.	Time	Best	Avg.	Time	
12	11	10	**11**	11	1.0	**11**	11	1.0	
13	27	21	**27**	25.9	22.2	**27**	26.3	14.3	
14	$\geqq$ 59	*DNR*	53	50.4	286.2	56	51.3	151.1	○

6.2.4 hamming シリーズ

hamming シリーズは，符号理論に基づく問題である．

パラメータ n と d をもつハミンググラフは，各々の長さ n の 2 進ベクトルに対して，1 つの節点をもっている．2 つの節点の隣接する必要十分条件は，その節点が対応するビットベクトルのハミング距離が少なくとも d 離れていることである．

また，hamming n-2 のグラフは，大きさ 2^{n-1} の最大独立集合をもつ．

表 6.6 DIMACS ベンチマークグラフ (hamming) に対する実験結果

Data	Opt.	CBH	GA			New			
			Best	Avg.	Time	Best	Avg.	Time	
15	32	**32**	**32**	32	0.0	**32**	32	0.0	
16	4	**4**	**4**	4	0.0	**4**	4	0.0	
17	128	**128**	**128**	128	1.0	**128**	128	1.0	
18	16	**16**	**16**	16	2.0	**16**	16	1.2	
19	512	**512**	**512**	512	12.1	**512**	512	6.1	
20	$\geqq 40$	35	36	34.7	39.2	**= 40**	36.3	27.8	◎

6.2.5 san シリーズ

san シリーズは，頂点被覆問題に基づく問題である．

グラフ $G = (V, E)$ の頂点被覆とは，$uv \in E$ なら $u \in A$ または $v \in A$ であるような集合 $A \subseteq V$ のことで，サイズ $|A|$ で表す．

頂点被覆問題は，グラフ G が与えられたとき，G の頂点被覆でサイズが最小のもの，つまり $|A|$ が最小のものを求める問題である．

また，san シリーズのグラフは，グラフが生成された段階ですでに最大独立集合のサイズが分かっている．

6.2.6 sanr シリーズ

sanr シリーズは，san シリーズと同様に頂点被覆問題に基づく問題である．

sanr シリーズのグラフのサイズは，san シリーズと同程度である．また，san シリーズの

表 6.7 DIMACS ベンチマークグラフ (san) に対する実験結果

Data	Opt.	CBH	GA			New			
			Best	Avg.	Time	Best	Avg.	Time	
21	30	15	**30**	30	2.0	**30**	30	1.1	
22	18	12	**18**	16.9	1.8	**18**	18	1.2	△
23	70	46	**70**	70	1.0	**70**	70	1.0	
24	60	36	**60**	60	1.0	**60**	60	1.0	
25	44	30	**44**	38.6	1.4	**44**	42.5	1.0	
26	13	8	**13**	12.7	5.5	**13**	12.6	5.0	
27	40	20	**40**	40	7.4	**40**	40	6.3	
28	30	15	**30**	28.1	6.0	**30**	26.5	5.4	
29	22	14	**22**	17.4	4.8	**22**	17.4	4.4	
30	100	50	**100**	100	4.1	**100**	100	3.2	
31	15	8	**15**	10.6	18.6	**15**	11.0	13.9	

グラフとは違い，sanr シリーズのグラフはランダムに生成されており，生成された段階では最大独立集合のサイズは分かっていない．

表 6.8 DIMACS ベンチマークグラフ (sanr) に対する実験結果

Data	Opt.	CBH	GA			New			
			Best	Avg.	Time	Best	Avg.	Time	
32	18	**18**	**18**	17.4	1.9	**18**	17.9	1.2	
33	≧ 42	41	**= 42**	41.2	1.5	**= 42**	41.2	1.2	
34	13	12	**13**	12.3	5.7	**13**	12.2	4.9	
35	≧ 21	20	**= 21**	19.5	5.7	**= 21**	20.3	5.2	

6.2.7 brock シリーズ

brock シリーズは，グラフ中の最大独立集合を隠すように (できるだけ分からなくするように) 作られた問題である．

brock シリーズのグラフでは，グラフ中にある最大独立集合の大きさは，節点数，枝数に

対して，小さい．

表 6.9 DIMACS ベンチマークグラフ (brock) に対する実験結果

Data	Opt.	CBH	GA			New			
			Best	Avg.	Time	Best	Avg.	Time	
36	21	20	**21**	20.2	1.9	**21**	20.1	1.5	
37	12	**12**	**12**	11.8	1.8	**12**	11.9	1.3	
38	15	14	**15**	14.8	1.9	**15**	14.5	1.4	
39	17	16	**17**	15.9	2.0	**17**	16.1	1.2	
40	27	23	24	22.9	6.3	25	23.7	5.0	○
41	29	24	24	22.9	6.0	24	23.5	4.9	
42	31	23	24	23.1	6.5	**31**	24.4	4.3	◎
43	33	24	**33**	24.2	6.0	**33**	24.0	4.5	
44	23	20	19	18.7	18.8	20	19.3	17.2	○
45	24	19	20	18.8	19.0	20	19.1	15.7	
46	25	20	19	18.7	20.3	20	19.1	17.1	○
47	26	19	19	18.5	20.5	20	19.0	17.5	○

6.2.8 p_hat シリーズ

p_hat シリーズは，ある条件の下でランダムに生成されたグラフに対する問題である．

p_hat シリーズのグラフは，各節点に接続されている枝数が少ないことから，最大独立集合のサイズは大きくなっている．

6.2.9 MANN シリーズ

MANN シリーズは，スタイナー問題に基づく問題である．

スタイナー問題の最も一般的な問題は，多次元距離空間内に n 個の点を与えるとき，それら n 個の点すべてを連結する最短のグラフ (スタイナー最小木) を求める問題である．

表 6.10 DIMACS ベンチマークグラフ (p_hat) に対する実験結果

Data	Opt.	CBH	GA			New			
			Best	Avg.	Time	Best	Avg.	Time	
48	8	**8**	**8**	8	3.1	**8**	8	2.8	
49	25	**25**	**25**	25	3.5	**25**	25	2.8	
50	36	**36**	**36**	35.0	4.0	**36**	34.9	2.9	
51	9	**9**	**9**	9	7.4	**9**	9	6.0	
52	36	35	**36**	35.8	12.4	**36**	35.8	8.6	
53	≧ 49	**= 49**	**• 50**	48.9	11.9	**• 50**	49.1	8.7	
54	11	**11**	**11**	10.5	17.5	**11**	10.8	12.8	
55	44	**44**	**44**	43.8	24.2	**44**	43.8	16.2	
56	≧ 62	60	**= 62**	61.4	23.6	**= 62**	60.7	15.2	
57	10	**10**	**10**	10	25.0	**10**	10	17.1	
58	≧ 46	**= 46**	**= 46**	45.3	46.2	**= 46**	45.7	30.1	
59	≧ 65	**= 65**	**• 68**	65.3	50.0	**• 68**	64.7	33.1	
60	12	11	11	10.7	47.8	**12**	11.2	33.4	◎
61	≧ 63	**= 63**	**• 65**	63.6	118.1	**• 65**	64.0	79.0	
62	≧ 94	**= 94**	93	91.4	116.6	93	91.3	75.3	

表 6.11 DIMACS ベンチマークグラフ (MANN) に対する実験結果

Data	Opt.	CBH	GA			New			
			Best	Avg.	Time	Best	Avg.	Time	
63	16	**16**	**16**	16	0.0	**16**	16	0.0	
64	126	121	**126**	125.5	2.1	**126**	125.6	1.0	
65	345	336	343	341.9	18.5	344	343.0	5.0	○
66	≧ 1100	DNR	1098	1095.0	223.0	1098	1097.0	39.7	

6.2.10 実験 2 の結果

実験 2 の結果より，以下のことが言える．

1. 付録 C，D の右端欄に ◎ 記号を付した 3 個のデータで，マルチデコード GA では得られなかった Opt. 解が得られた．

2. 付録 C，D の右端欄に ○ 記号を付した 6 個のデータで，マルチデコード GA より良い解が得られた．
3. 付録 C，D の右端欄に △ 記号を付した 1 個のデータで，10 回の試行すべてにおいて Opt. 解が得られるようになった．
4. 付録 C，D の最下行の合計欄から，ベンチマーク全体において，マルチデコード GA より良い解が少ない処理時間で得られていることが分かる．

6.3 考察

マルチデコード GA と本手法を実験 1 で使用した keller5 のグラフを用いて，計算時間，収束速度，最適解を得る確率について考察する．

6.3.1 計算時間

付録 D から分かるように，本手法の平均処理時間の合計は 1008.8 秒であり，マルチデコード GA の平均処理時間の合計 1446.6 秒よりも短く，その減少率は，DIMACS ベンチマークグラフ全体において，約 30.3 %となり，本手法の有用性が認められる．ただし，この減少率の差は主に交叉時間とウイルス感染時間の差によるものと考えられる．すなわち，マルチデコード GA では，世代ごとに $M/2$ 個の染色体対を形成し，適当な交叉を施し，新しい染色体集団を形成している．しかし，本手法のウイルス感染では，一つの染色体を選び，ウイルスを決定し感染させる．これは，集団サイズが大きくなっても処理時間が変化しないことになる．図 6.1 に示す処理時間と適応度の関係からも明らかなように，本手法 (New) はマルチデコード GA(GA) に比べ短い時間で同様の探索が可能になっている．ただし，図 6.1 の適応度は，各 10 回の試行における適応度の最大値を時間ごとに平均したものである．

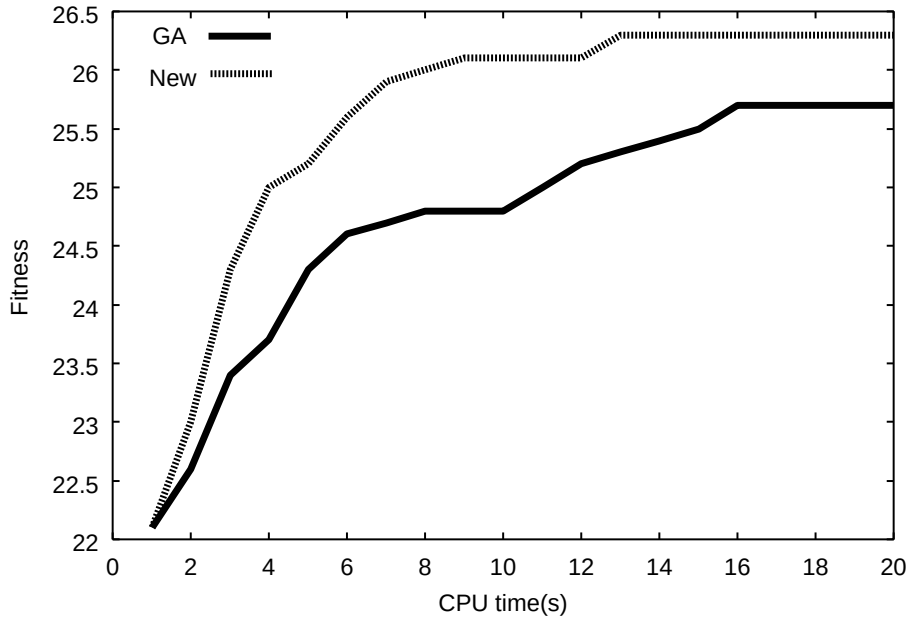


図 6.1　処理時間と適応度の関係 (keller5)

6.3.2　収束速度

マルチデコード GA および本手法における世代数と適応度の関係を，それぞれ図 6.2，図 6.3 に示す．これらの図で示す適応度は，各 10 回の実験での適応度の最大値，平均値，最小値を世代毎に平均したものである．初期世代の適応度の変化からマルチデコード GA よりも本手法の方が収束が速く，また，その収束した最大値から，本手法の方が探索能力が高いことが分かる．この収束と探索能力の差は，本手法のウイルス感染という遺伝子操作が，ある一つの解がもつ情報を集団全体に広く拡散させるという操作のために，マルチデコード GA よりも集団全体がある一つの解の周辺に集中しやすくなり，その結果として局所探索能力が高くなっていることに原因があると思われる．しかし，図 6.2 で示すマルチデコード GA の平均値は徐々に良い方向へと進み，最小値にはばらつきが見受けられるが，図 6.3 で示す本手法の平均値，最小値を見る限り，ほとんどばらつきがないことが分かる．これは，本手法のアルゴリズムが良い解は良い解へと短時間で収束し，悪い解はいつの世代にも存在していることを示唆している．

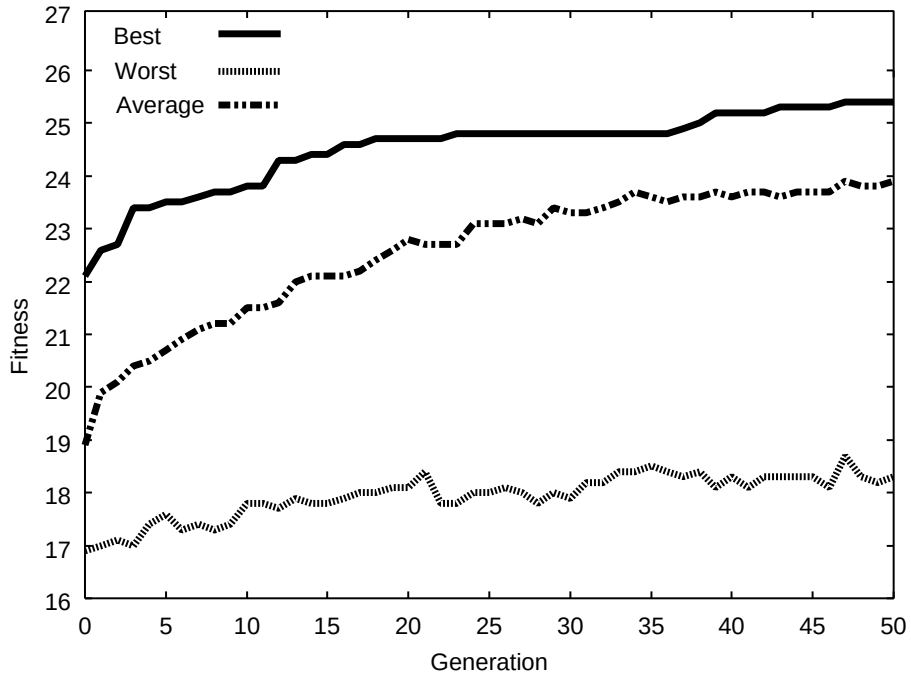


図 6.2　マルチデコード GA における適応度の変化 (keller5)

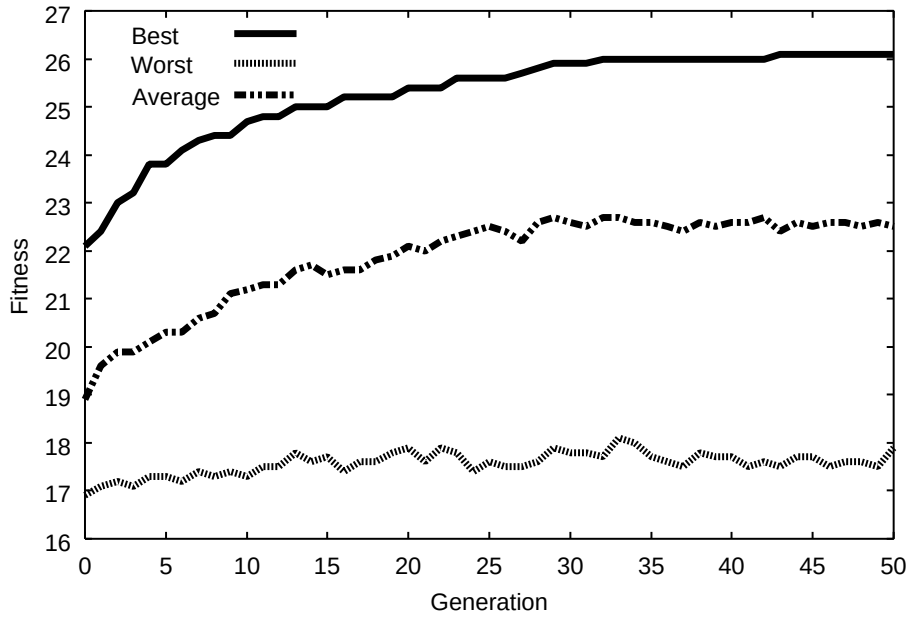


図 6.3　本手法における適応度の変化 (keller5)

6.3.3　最適解を得る確率

keller5 では，最適解の適応度が 27 であることが分かっている．マルチデコード GA では，この最適解を 10 回中 4 回得られている．本手法では，10 回中 5 回得られており，その確率は僅かながら上がっている．このことから，本手法の方が局所探索性に優れていると考えられる．

6.4　まとめ

実験の結果，すべての DIMACS ベンチマークグラフに対してマルチデコード GA より，同等あるいは良い解が得られた．しかし，最適解を得ることができなかったベンチマークグラフも幾つか存在する．このことは，本研究で提案したアルゴリズムにおいて十分改良の余地があることを示している．

次章では，以上の実験結果を踏まえた上での結論と今後の課題について述べる。

第 7 章

結論

本研究では，最大独立集合問題に対して，文献 [7] で報告されているマルチデコード GA の解法における交叉の有効性についての問題点に着目し，ウイルス進化論の考え方を導入し，“交叉の代わりにウイルス感染” を用いた遺伝的アルゴリズムを提案した．その結果，すべてのベンチマークグラフに対してマルチデコード GA より，同等あるいは良い解が得られた．しかしながら，既知の最適解あるいは最良解を求めることができなかったベンチマークグラフが幾つか残っている．このことから，提案したウイルス感染手法には改良の余地がある．今後の研究の方向として，次の 2 点が挙げられる．

1．新たなウイルス感染手法の提案

より探索範囲を広げることができるようなウイルス感染手法を考案する必要がある．

2．新たな突然変異の提案

本研究のアルゴリズムでは，最も単純な突然変異しか採用していないため，ウイルス感染手法に適した突然変異を工夫することでより良い解を得るための手助けになると思われる．

謝辞

本論文は，著者が 2001 年 3 月から 2003 年 2 月までの高知工科大学大学院工学研究科基盤工学専攻情報システム工学コース在学中に，同コースの坂本研究室において行った研究の成果を記したものである．

はじめに，著者の大学院生活が有意義に送ることができたことに感謝致します．

次に，GA から本論文，プログラムなど種々の書式に至るまで，御指導・御教示を賜わった坂本 明雄教授，輪講から研究活動まで種々の面で御助言を賜わった福本 昌弘助教授に深く感謝致します．

また，研究室活動において，同輩の友池 貴之氏，学部 4 年生の赤間 寛氏，河内 友彦氏，河野 兼祐氏，西村 章氏に種々の面で支えになって頂いたことに感謝致します．

そして，研究室の学部 3 年生諸君の暖かい気配りに感謝すると共に，これから迫りくる研究活動を頑張って下さい．

さらに，研究活動中に，種々の相談にのってもらい，いろいろな面でアドバイスをくれた同コースの橋本 正和氏，平山 正治氏，中原 昌樹氏，秋山 由佳氏，舟橋 釈仁氏に感謝致します．

最後に，本論文を審査して下さった坂本 明雄教授，竹田 史章教授，福本 昌弘助教授並びに，著者が本大学院入学時から今まで過ごしやすい環境を整えて頂いた情報システム工学コースおよび情報システム工学科の諸先生方に感謝の意を表します．

参考文献

[1] I. M. Bomze, M. Budinich, P. M. Pardalos and M. Pelillo, “The maximum clique problem”, *Handbook of Combinatorial Optimization*, vol. 4, D.-Z. Du and P. M. Pardalos (Eds.), Kluwer Academic Publishers, Boston, MA, 1999.

[2] 北野 宏明，遺伝的アルゴリズム，産業図書，1993.

[3] A. S. Murthy, G. Parthasarathy and V. U. K. Sastry, “Clique finding − A genetic approach”, *Proc. 1st IEEE Conf. on Evolutionary Computation*, pp. 18–21, 1994.

[4] T. N. Bui and P. H. Eppley, “A hybrid genetic algorithm for the maximum clique problem”, *Proc. 6th International Conf. on GAs*, pp.478–484, 1995.

[5] L. E. Gibbons, D. W. Hearn and P. M. Pardalos, “A continuous based heuristic for the maximum clique problem”, *Reseach Report 94-9*, Department of Industrial & Systems Engineering, University of Florida, 1994.

[6] A. Sakamoto, X. Liu and T. Shimamoto, “A genetic approach for maximum independent set problems”, *IEICE Trans. Fundamentals*, vol. E80-A, no. 3, pp. 551–556, Mar. 1997.

[7] 島本 隆， 來山 征士， 橋本 学， 坂本 明雄，“マルチデコード GA を用いた最大独立集合問題の解法”，信学論 (A)，vol. J83-A，no. 12，pp. 1505–1511，Dec. 2000.

[8] 中原 英臣， 佐川 峻，ウイルス進化論，泰流社，1989.

[9] 中谷 直司， 金杉 昭徳， 近藤 邦雄，“ウイルス進化論に基づく進化型アルゴリズム”，情報処理学会論文誌，vol. 40，no. 5，pp. 2346–2355，May 1999.

[10] 登伸一，坂本明雄，島本隆，“ウイルス進化型 GA による最大独立集合問題の解法”，電気関係学会四国支部連合大会講演論文集，p. 5，Oct. 2002.

[11] 登伸一，坂本明雄，島本隆，“ウイルス感染を用いた最大独立集合問題の進化的解法”，第 25 回情報理論とその応用シンポジウム，vol. 2，pp. 767–770，Dec. 2002.

付録 A

グラフ名とその対応番号 1

	Data	Nodes	Edges	Density
1	c-fat200-1	200	1534	7.7
2	c-fat200-2	200	3235	16.3
3	c-fat200-5	200	8473	42.6
4	c-fat500-1	500	4459	3.6
5	c-fat500-2	500	9139	7.3
6	c-fat500-5	500	23191	18.6
7	c-fat500-10	500	46627	37.4
8	johnson8-2-4	28	210	55.6
9	johnson8-4-4	70	1855	76.8
10	johnson16-2-4	120	5460	76.5
11	johnson32-2-4	496	107880	87.9
12	keller4	171	9435	64.9
13	keller5	776	225990	75.2
14	keller6	3361	4619898	81.8
15	hamming6-2	64	1824	90.5
16	hamming6-4	64	704	34.9
17	hamming8-2	256	31616	96.9
18	hamming8-4	256	20864	63.9
19	hamming10-2	1024	518656	99.0
20	hamming10-4	1024	434176	82.9
21	san200_0.7_1	200	13930	70.0
22	san200_0.7_2	200	13930	70.0
23	san200_0.9_1	200	17910	90.0
24	san200_0.9_2	200	17910	90.0
25	san200_0.9_3	200	17910	90.0
26	san400_0.5_1	400	39900	50.0
27	san400_0.7_1	400	55860	70.0
28	san400_0.7_2	400	55860	70.0
29	san400_0.7_3	400	55860	70.0
30	san400_0.9_1	400	71820	90.0
31	san1000	1000	250500	50.2
32	sanr200_0.7	200	13868	69.7
33	sanr200_0.9	200	17863	89.8
34	sanr400_0.5	400	39984	50.1
35	sanr400_0.7	200	55869	70.0

付録 B

グラフ名とその対応番号 2

Data		Nodes	Edges	Density
36	brock200_1	200	14834	74.5
37	brock200_2	200	9876	49.6
38	brock200_3	200	12048	60.5
39	brock200_4	200	13089	65.8
40	brock400_1	400	59723	74.8
41	brock400_2	400	59786	74.9
42	brock400_3	400	59681	74.8
43	brock400_4	400	59765	74.9
44	brock800_1	800	207505	64.9
45	brock800_2	800	208166	65.1
46	brock800_3	800	207333	64.9
47	brock800_4	800	207643	65.0
48	p_hat300-1	300	10933	24.4
49	p_hat300-2	300	21928	48.9
50	p_hat300-3	300	33390	74.4
51	p_hat500-1	500	31569	25.3
52	p_hat500-2	500	62946	50.5
53	p_hat500-3	500	93800	75.2
54	p_hat700-1	700	60999	24.9
55	p_hat700-2	700	121728	49.8
56	p_hat700-3	700	183010	74.8
57	p_hat1000-1	1000	122253	24.5
58	p_hat1000-2	1000	244799	49.0
59	p_hat1000-3	1000	371746	74.4
60	p_hat1500-1	1500	284923	25.3
61	p_hat1500-2	1500	568960	50.6
62	p_hat1500-3	1500	847244	75.4
63	MANN_a9	45	918	92.7
64	MANN_a27	378	70551	99.0
65	MANN_a45	1035	533115	99.6
66	MANN_a81	3321	5506380	99.9

付録 C

実験結果 1

Data	Opt.	CBH	GA			new			
			Best	Avg.	Time	Best	Avg.	Time	
1	12	**12**	**12**	12	3.0	**12**	12	3.0	
2	24	**24**	**24**	24	4.0	**24**	24	4.0	
3	58	**58**	**58**	58	3.0	**58**	58	2.0	
4	14	**14**	**14**	14	18.0	**14**	14	17.0	
5	26	**26**	**26**	26	32.0	**26**	26	31.0	
6	64	**64**	**64**	64	50.0	**64**	64	48.3	
7	$\geqq$ 126	**= 126**	**= 126**	126	33.6	**= 126**	126	32.0	
8	4	**4**	**4**	4	0.0	**4**	4	0.0	
9	14	**14**	**14**	14	0.0	**14**	14	0.0	
10	8	**8**	**8**	8	1.0	**8**	8	0.0	
11	16	**16**	**16**	16	4.0	**16**	16	3.0	
12	11	10	**11**	11	1.0	**11**	11	1.0	
13	27	21	**27**	25.9	22.2	**27**	26.3	14.3	
14	$\geqq$ 59	DNR	53	50.4	286.2	56	51.3	151.1	○
15	32	**32**	**32**	32	0.0	**32**	32	0.0	
16	4	**4**	**4**	4	0.0	**4**	4	0.0	
17	128	**128**	**128**	128	1.0	**128**	128	1.0	
18	16	**16**	**16**	16	2.0	**16**	16	1.2	
19	512	**512**	**512**	512	12.1	**512**	512	6.1	
20	$\geqq$ 40	35	36	34.7	39.2	**= 40**	36.3	27.8	◎
21	30	15	**30**	30	2.0	**30**	30	1.1	
22	18	12	**18**	16.9	1.8	**18**	18	1.2	△
23	70	46	**70**	70	1.0	**70**	70	1.0	
24	60	36	**60**	60	1.0	**60**	60	1.0	
25	44	30	**44**	38.6	1.4	**44**	42.5	1.0	
26	13	8	**13**	12.7	5.5	**13**	12.6	5.0	
27	40	20	**40**	40	7.4	**40**	40	6.3	
28	30	15	**30**	28.1	6.0	**30**	26.5	5.4	
29	22	14	**22**	17.4	4.8	**22**	17.4	4.4	
30	100	50	**100**	100	4.1	**100**	100	3.2	
31	15	8	**15**	10.6	18.6	**15**	11.0	13.9	
32	18	**18**	**18**	17.4	1.9	**18**	17.9	1.2	
33	$\geqq$ 42	41	**= 42**	41.2	1.5	**= 42**	41.2	1.2	
34	13	12	**13**	12.3	5.7	**13**	12.2	4.9	
35	$\geqq$ 21	20	**= 21**	19.5	5.7	**= 21**	20.3	5.2	

付録 D

実験結果 2

Data	Opt.	CBH	GA			new			
			Best	Avg.	Time	Best	Avg.	Time	
36	21	20	**21**	20.2	1.9	**21**	20.1	1.5	
37	12	**12**	**12**	11.8	1.8	**12**	11.9	1.3	
38	15	14	**15**	14.8	1.9	**15**	14.5	1.4	
39	17	16	**17**	15.9	2.0	**17**	16.1	1.2	
40	27	23	24	22.9	6.3	25	23.7	5.0	○
41	29	24	24	22.9	6.0	24	23.5	4.9	
42	31	23	24	23.1	6.5	**31**	24.4	4.3	◎
43	33	24	**33**	24.2	6.0	**33**	24.0	4.5	
44	23	20	19	18.7	18.8	20	19.3	17.2	○
45	24	19	20	18.8	19.0	20	19.1	15.7	
46	25	20	19	18.7	20.3	20	19.1	17.1	○
47	26	19	19	18.5	20.5	20	19.0	17.5	○
48	8	**8**	**8**	8	3.1	**8**	8	2.8	
49	25	**25**	**25**	25	3.5	**25**	25	2.8	
50	36	**36**	**36**	35.0	4.0	**36**	34.9	2.9	
51	9	**9**	**9**	9	7.4	**9**	9	6.0	
52	36	35	**36**	35.8	12.4	**36**	35.8	8.6	
53	≧ 49	**= 49**	**• 50**	48.9	11.9	**• 50**	49.1	8.7	
54	11	**11**	**11**	10.5	17.5	**11**	10.8	12.8	
55	44	**44**	**44**	43.8	24.2	**44**	43.8	16.2	
56	≧ 62	60	**= 62**	61.4	23.6	**= 62**	60.7	15.2	
57	10	**10**	**10**	10	25.0	**10**	10	17.1	
58	≧≧ 46	**= 46**	**= 46**	45.3	46.2	**= 46**	45.7	30.1	
59	≧≧ 65	**= 65**	**• 68**	65.3	50.0	**• 68**	64.7	33.1	
60	12	11	11	10.7	47.8	**12**	11.2	33.4	◎
61	≧≧ 63	**= 63**	**• 65**	63.6	118.1	**• 65**	64.0	79.0	
62	≧≧ 94	**= 94**	93	91.4	116.6	93	91.3	75.3	
63	16	**16**	**16**	16	0.0	**16**	16	0.0	
64	126	121	**126**	125.5	2.1	**126**	125.6	1.0	
65	345	336	343	341.9	18.5	344	343.0	5.0	○
66	≧ 1100	DNR	1098	1095.0	223.0	1098	1097.0	39.7	
total	4171	2742	4125	4067.3	1446.6	4145	4082.8	1008.8	