多重プログラム型人工生命 Tierra の進化メカニズムに関する研究

菊地慶仁*・清水祐輔**・桃内佳雄***

Research for evolution mechanism in multi program artificial life "Tierra"

Yoshihito KIKUCHI*, Yusuke SHIMIZU** and Yoshio MOMOUCHI***

概　要

本研究は，外的環境と関連を持ち，生物として多様な振るまいを可能とさせる人工生命の開発を目的としている。このために生物の構造を，自己複製プログラムのみを持つ形式から，外的環境と相互に関連する多目的なプログラムを持つ形式へと，拡張を試みる。特に今回は，複数のプログラムを持つ生物の淘汰に関する問題点を指摘し，生物の種類もしくは自己複製回数のどちらかを優先することを可能とする淘汰の方式を提案し，その有効性を実験によって確認する。

1．人工生命 Tierra の特徴及び問題点

1.1　特徴

Tierra は，Tom Ray[1]によって生物の進化プロセスを観察するために作られたシミュレーションプログラムである。以下では，幾つかの項目に従って，Tierra の特徴について述べる。

＊北海学園大学工学部電子情報工学科（札幌市中央区南 27 条西 11 丁目 1－1）
＊Department of Electronics and Information Engineering, Faculty of Engineering, Hokkai-Gakuen University (1-1, South 26, West 11, Chuo-ku, Sapporo)
＊＊北海学園大学大学院工学研究科（同上）
＊＊Electronics and Information Engineering, Graduate Course of Engineering, Hokkai-Gakuen University
＊＊＊北海学園大学工学部電子情報工学科（同上）
＊＊＊Department of Electronics and Information Engineering, Faculty of Engineering, Hokkai-Gakuen University

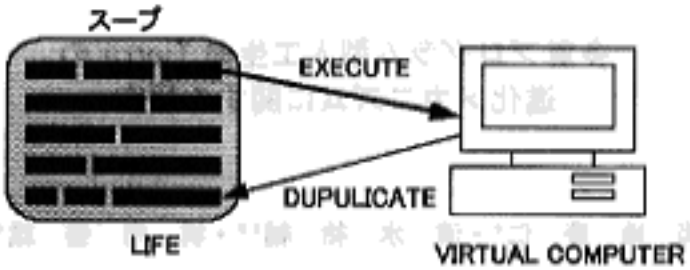

図1 Tierra の構造と基本的な実行メカニズム

実行メカニズム

Tierra では，生物は自己複製が可能なプログラムとして実装されている。各生物は，仮想コンピュータ用に設計された命令セットを用いたプログラムで，スープと呼ばれるメモリ空間中に置かれる。スープ中の各生物は，仮想コンピュータから CPU 時間（実行命令回数に影響）を割り当てられることによってプログラムを実行して自己複製を繰り返しながら，限りある CPU 時間とメモリ領域を互いに奪い合う（図1）。

Tierra は，自分または他の生物のプログラムの一部をサブルーチンのように他の生物から使用することも可能である［2］［3］。この能力と突然変異による命令の変化によって，別の生物のプログラムを利用する寄生種等の亜種が生み出される。

プログラムの構造とオリジナル生物

生物のプログラムには，以下の2つの特徴がある。

1．32個のアセンブラ言語レベルの命令を用いた命令セット

Tierra 用のプログラムは，高級言語ではなく，16進コードで表現された仮想コンピュータ用のマシンコードを用いている。実際のスープ中には，このコードが直接格納される。以下に［3］より，初期の Tierra で用いられた命令セットを示す。

命令	内容
0x00	no operation
0x01	no operation
0x02	flip low order bit of cx, $cx=1$
0x03	shift left cx register, $cx \ll =1$
0x04	set cx register to zero, cx=0
0x05	if cx==0 execute next instruction
0x06	subtract bx from ax, cx=ax−bx
0x07	subtract cx from ax, ax=ax−cx

0x08　increment ax, ax＝ax＋1
0x09　increment bx, bx＝bx＋1
0x0a　decrement cx, cx＝cx－1
0x0b　increment cx, cx＝cx＋1
0x0c　push ax on stack
0x0d　push bx on stack
0x0e　push cx on stack
0x0f　push dx on stack
0x10　pop top of stack into ax
0x11　pop top of stack into bx
0x12　pop top of stack into cx
0x13　pop top of stack into dx
0x14　move ip to template
0x15　move ip backward to template
0x16　call a procedure
0x17　return from a procedure
0x18　move cx to dx, dx＝cx
0x19　move ax to bx, bx＝ax
0x1a　move instruction at address in bx to address in ax
0x1b　address of nearest template to ax
0x1c　search backward for template
0x1d　search forward for template
0x1e　allocate memory for daughter cell
0x1f　cell division

2．相補的なパターン（テンプレート）によるアドレッシング

通常のアセンブラ言語では，ジャンプもしくはサブルーチンコールを行うアドレスなどは実際のメモリアドレス(もしくはマクロ宣言されたアドレス)を指定する必要があるが，Tierra の仮想コンピュータ上にはアドレスの概念は無い。このために，分岐，ジャンプ，サブルーチンコールなどのアドレスを指定するために，2種類の non-operation 命令を組合せたパターン（テンプレート）による指定を用いる。

相補的なパターンによるアドレッシングとは，non-operation 命令（nop_0，nop_1）の組み合わせでテンプレートを作り，ジャンプ等の命令の実行のときにアドレスを指定する方式である。図2にテンプレートによるアドレッシングの例を示す。始めにジャンプ命令が示され，

その次に4つのnop命令の並びで飛び先のアドレスを示すためのテンプレート（ここでは0011）が示されている。実際の飛び先は，このテンプレートの0/1を反転させた相補的なテンプレートとなる。すなわち，この場合のjmp命令は，引き続く1100のパターンの次までジャンプすることを指定している。

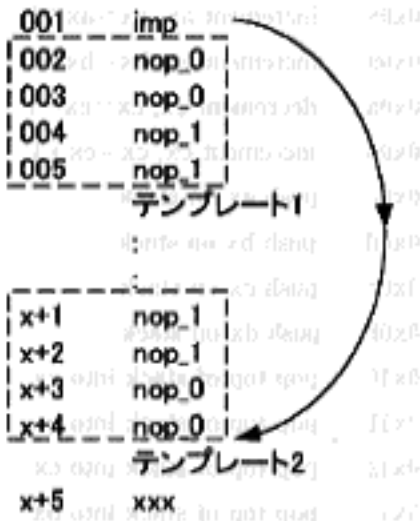

図2　テンプレートによるアドレッシング

Tom Rayによるシミュレーション[1]では，先祖種と呼ばれる80個の命令によって構成された1種類の生物から自己複製が開始される。このプログラムは，

- プログラムの先頭と終端とを示すテンプレートの間の命令数から自分の命令長求め，その結果を特定のレジスタにスタックする。
- スープ中で空いているメモリ領域を見つける。
- 先頭アドレスから自分自身のプログラムの内容を，空いている領域にコピーする。
- プログラムの先頭に戻って，実行を繰り返す。

といった内容になっており，スープに空き領域がある限り自己複製を続ける。

突然変異

Tierraでは，自己複製中の突然変異によって，プログラムコードが書き換わり，1つのプログラムから多様なプログラムが生まれる。

生物の進化は，次の変化によって引き起こされる。

1．スープ中のビットの反転（宇宙線によって起こる突然変異に相当）

2．命令の実行内容の変化

この2つの変化は一定の確率で起こり，複製中にオリジナルと異なるプログラムを持つ生物が生成される可能性が生じる。特に，前述のテンプレート中でビット反転が起こると，オリジナルの生物以外の別の生物のテンプレートをサブルーチン呼び出しの先として指定したために，他の生物を利用して自分の命令列を複製させる寄生種などが結果として生まれることになる。しかしながら，自己複製可能な完全な形のプログラムから，同じように自己複製可能な完全な形のプログラムに突然変異することは非常に稀であり，大半の変異種では1度の実行で停止してしまう不完全なプログラムとなってしまうことが多い。

淘汰

Tierraにおける淘汰は，各生物が複製を繰り返すことによって生物数が増えスープが80%

使用された段階で，リーパーと呼ばれるルーチンが起動されることで起きる。このため，リーパーキューと呼ばれるデータ構造が用意されている。キューには生物が生まれた順番に並んでおり，リーパーは，キューの先頭の生物のメモリ領域を開放することによって，その生物を削除することで淘汰を行う。このキューの中では，生物が複製などの特定の命令の実行に失敗すると順番が上がり，別の特定の命令の実行に成功すると順番が下がる。このメカニズムによって，プログラムに問題のある生物は先に淘汰されることになる。

1.2 Tierra における問題点

Tierra における進化とは，自己複製中に発生する突然変異によって，結果として多様な種類のプログラムを持つ生物が生まれることである。このような進化を実現するためには，自己複製が可能な生物が多くスープ中に存在する必要がある。しかし，この状態を実現するにはオリジナルの Tierra にはいくつかの問題点がある。以下に種の多様化が促進されない Tierra の淘汰における問題点を指摘する [4]。

自己複製に関しての問題点

Tierra のプログラムは，自身の自己複製プログラムを自己複製することしか行なえない。特に生物として見た場合に，次の項目が必要になると考えられる。

1．外部環境の対応

2．自身の内部状態を保持するメカニズム

3．上記の 2 つのパラメータを扱う，複数の目的毎の自己複製以外のルーチン

淘汰に関しての問題点

1．リーパーによる淘汰に関する問題点

突然変異で変化した生物のプログラムの中で，自己複製が可能なプログラムを持つ生物は少ない。リーパーによる淘汰は，生物が複製できたかどうかには殆ど関係していなかったため，複製ができなかった生物を優先的に淘汰することができない。従って，自己複製が可能な生物が生まれにくいことになる。

2．CPU 時間に関する問題点

オリジナルの Tierra システムでは，生物に割り当てる CPU 時間は生物のサイズ(命令数)の冪乗に比例する形になっている。この冪数 n を $(0<n<\infty)$ の範囲で変えることによって，大きなサイズの生物が有利に，もしくは小さな生物が有理に CPU 時間を割り当てられる。しかしこれでは，サイズが小さい生物か大きい生物のどちらかにしか有利な環境を作ることができない。サイズが小さい場合，単純に自己複製が効率的に行われると解釈することもできるが，寄生種の方がより小さなサイズとなる。しかしながら，寄生種は寄生する相手の宿主生物が淘汰されると，その宿主を使っていた全ての寄生種も淘汰されることとな

る。一概にサイズが小さい生物を優先させ寄生種ばかりとなることは，生態系としては不安定になる可能性を持つことになる。

3．突然変異確率に関する問題点

Tom Ray の論文 [1] では，プログラムに影響を与える突然変異は，すべての生物に対して平等の確率で起こっていた。しかし，自己複製が不可能な生物に対する突然変異確率が低いと，自己複製が可能になるまで時間がかかってしまう。

従って，自己複製が可能な種から子孫を多様化させるためには，生まれる生物数，生物の種類が増加するように従来のリーパーキューによる淘汰ではなく，新たな淘汰の方式を提案すること必要となる。次章では，これらの問題点に対する解決策を提案する。

2．多重プログラム型 Tierra の提案

前節であげた問題点に対して，本報告では，主に次の 2 点を提案する。

1．複数のプログラムを持った多重プログラム型 Tierra の基本構造

2．上記多重プログラム型 Tierra の並列プログラム構造における，より洗練された淘汰メカニズム

以下では，それぞれの特色について述べる。

2.1　基本構造

生物中の自己複製プログラムと並列プログラム，及び外的環境の関係を図 3 に示す。自己複

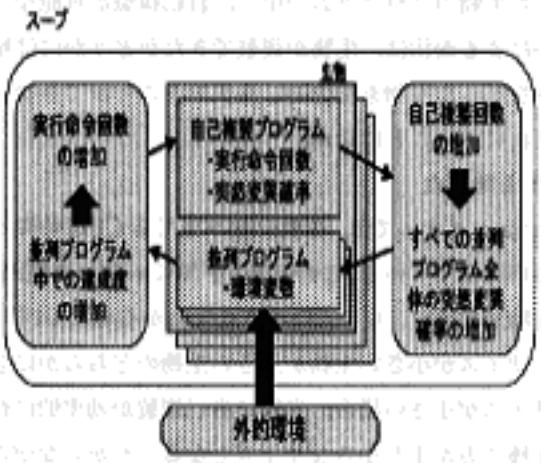

図 3　生物と外的環境との関係及び共生のイメージ

製プログラムと並列プログラムは並行動作し，自己複製によって一緒に複製される。並列プログラムは，自己複製プログラムと基本的には同じレベルの言語が用いられるが，プログラムの目的に応じて自己複製プログラムとは別の命令セットを持つことができる。例えば次のような命令が考えられる。

1．空間移動を行う問題に対するプログラムのための移動命令群

2．生物の外的な環境変数を獲得するプログラムのための命令群

3．具体的な動作は行わずに，生物の静的な状態変数を蓄積するプログラムのための命令群

このように自己複製プログラムと別の目的のプログラムを組合せることは，特定タンパク質を生成する遺伝子を大腸菌に注入して大量生産を行う生物工学に類似している。

Tierra では，プログラム自体の進化と他の生物の優秀なプログラムを利用できる進化が可能であり，複数のプログラムを持たせた場合でも，これらと同等の進化が期待できる。すなわち，自己複製以外の並列プログラムの各々が進化し，寄生などを繰り返すことで，より進んだ生物が発生することや，また複数の生物が全体として一つの生物として共生しながら発達することが考えられる。

自己複製プログラムと並列プログラムは以下のような影響を相互に与えている。

1．自己複製が可能な場合，その複製回数によりプログラム全体の突然変異確率が増減する。

2．並列プログラム中での達成度によって，実行命令回数と突然変異確率が増減する。

2.2　淘汰メカニズム

多様な種を生むためには，自己複製が可能な種を多く保存し，さらに自己複製時の突然変異によって，生まれる生物の種類も多くしなければならない。淘汰は [2] での方式を複数プログラムに拡張し，スープが 80%使用された時に，全生物におけるDの値が一番大きな生物を除去することで行う。この判別値におけるSは生物の年齢を表し，R_1，R_2は複製に要した命令数の累計である。複製回数が多く，その中で年齢が若い生物がより生き残りやすくなる。

1．淘汰に関する提案

より多くの複製可能なプログラムを持つ生物がスープ中に残れるように，本報告ではリーパーキューを廃止し，複製回数による報酬及び命令実行の失敗回数によるペナルティによって淘汰を行う。これによって，複製が可能な生物がスープ中に多数存在する可能性が高くなる。淘汰の判別値は式 1 によって求められる。

$$D=S-(R_1+R_2)+P \tag{1}$$

ここで

S：現在の命令数－生物が生まれた命令数

R_1：自分と同じ命令列の複製回数×w_1

R_2：自分と異なる命令列の複製回数×w_2

P　：命令の実行に失敗した回数×w_3

（w_1，w_2，w_3はウェイト）

2．CPU時間に関する提案

生物のプログラムサイズに単純に依存していた従来の形式を，一回に実行できる命令の数，つまり割り当てられるCPU時間が生物が持つ並列プログラムの実行結果によって変化するようにする。これによって，CPU時間がサイズに依存しなくなり，いろいろなサイズの生物が出てくることが期待できる。

3．突然変異確率に関する提案

突然変異確率も同様に，生物の並列プログラムの実行結果によって変化するようにし，プログラムが変化しやすい生物と，変化しにくい生物が生まれるようにする。これによって，あまり良いプログラムを持っていない生物でも，実行結果による突然変異確率が高い場合は，プログラムにより多くの変化が起こり，良いプログラムに変化する可能性が生じると期待される。また生物の存在する環境が悪い場合，突然変異確立が上がり環境適応性が高くなるモデルなども考えられる。

今回の提案におけるオリジナルとの変更点を表1にまとめる。

表1　オリジナルと本提案との変更点

	オリジナル	提案するモデル
報酬	特定の命令実行に成功するたびにリーバーキュー内の順番が下降	自分と同じ命令列及び違う命令列の複製回数に応じて報酬を与える
ペナルティ	特定の命令実行に失敗するたびにリーバーキュー内の順番が上昇	特定の命令実行に失敗した回数に応じてペナルティを与える
淘汰される生物	リーバーキューの先頭にいる生物	スープ内にいる全生物に対して淘汰の判別値を計算して値が大きい生物を淘汰する
内部情報	なし	並列プログラムの実行結果によって突然変異確率と，一回に実行できる命令回数に影響を与える

3．実　験

本提案の有効性を確かめるためにTierraシステムを作成し，淘汰にリーバーキューを用いた場合と，今回の提案を加えた場合について実験を行った。大きくは次の2点に関して実験を行う。

●並行プログラムが存在せず自己複製しか行わない状態での，淘汰方式の有効性の確認

●並列プログラムが動作している状況での淘汰方式の有効性の確認

開発環境は，OS：Windows 98，コンパイラ：Visual C++ 6.0，CPU：Pentium III 600 MHz で，3500 万命令を実行するのに約 7 時間かかる[1]。

3.1 自己複製のみでの淘汰方式の有効性の確認

実験では，3500 万命令実行時で以下の点について比較を行うことで，本提案の有効性を確認する。

●生まれた生物数

●生物の種類

●生まれた生物中の自己複製が可能だった生物数

●生物の長さの変化

また，本提案における判別値のR_1，R_2，Pのウェイトを変えることによって，ウェイトの良好な組み合わせを探る。

スープ中の全生物が自分を一回複製するためには最低 4 ～ 5 万命令実行が必要であることがわかっているので，w_1が大きすぎると生物がいつまでも淘汰されないで残り，逆に少なすぎると複製できる生物がスープ中に残らなくなると考えられる。よってw_1＝20000，25000，30000 とした。w_2に関しては，複製のみに成功しているという意味合いがある。今回は，自己複製能力を重要視したのでw_1との差をつけるためw_1の半分とした。w_3に関しては，このウェイトを大きくすることで欠陥のあるプログラムを排除することが可能だと思われるが，あまり大きすぎるとプログラムに制約を課すことになると考えられるので 3000 もしくは 5000 とした。

内部情報で指定できる突然変異確率は，オリジナルの Tierra の確率の 1/2，1，2 倍の 3 つの値を取るようにした。同じく内部情報で指定できる一回に実行できる命令回数については，2，3，4 回の 3 つの値を取るようにした。

表 2　実験系列ごとのウェイト及びパラメータの値

系　列	w_1	w_2	w_3	突然変異確率	命令実行回数
203_av	20,000	10,000	3,000	1/2，1，2	2，3，4
205_av	20,000	10,000	5,000	1/2，1，2	2，3，4
253_av	25,000	12,500	3,000	1/2，1，2	2，3，4
255_av	25,000	12,500	5,000	1/2，1，2	2，3，4
303_av	30,000	15,000	3,000	1/2，1，2	2，3，4
305_av	30,000	15,000	5,000	1/2，1，2	2，3，4

[1] 今回使用したプログラムは，論文等を参考にして独自に開発したもので，Tom Ray 氏が作成したオリジナルのシステムとは同一ではないが動作については変わらないことを付記しておく。

3.2 自己複製と並行プログラムが動作する状況での淘汰方式の有効性の確認

複数プログラムを持つ場合での前述の年齢と複製回数の判別値中での重要度に関して実験を行う。今回は，移動プログラムを生物に持たせ，例題として山登りを考える。ただし，移動プログラムが直接実行されるのではなく，ある既定値の報酬（実行命令回数の増減）を与えられたとして行う。本報告では，淘汰の判別値Dにおけるウェイトの値を表3のように設定し，移動プログラムからの報酬有り，無しの場合の2つで1億命令実行時における生物数，生物の種類等の比較を行う。

表3　実験で使用する条件

移動アルゴリズムからの報酬	w_1	w_2	w_3
有り	30，50，70，90%	w_1 の半分	5000
無し	30，50，70，90%	w_1 の半分	5000

ここでw_1をw_2の倍としたのは自己と同一のプログラムへの複製を優先したためで，w_3を5000に固定しているのは［4］においてこの値が最適だったためである。

3.3 実験結果1

パラメータを変えて実験を行い，3500万命令実行時までの結果についての考察を行う。

生まれた生物数では，図4よりオリジナルに比べて今回の提案のほうがおよそ3倍になっていることがわかる。また生物の種類に関しても，図5より生物数と同様におよそ3倍になっている。

両方のグラフの傾きより，実行命令数が増加すると生物数，生物の種類ともオリジナルとの差が広がる傾向がある。

次に複製が可能な生物数を見てみると，図6より生まれた生物数に対して全ての条件において約3分の1になっている。このことから，複製が可能な生物の数は生まれる生物数が多ければ多いほど増えるということになる。

ウェイトを変化させた場合は，図4，図5より，わずかではあるがw_1＝20000の時に良い結果を出している。これは，報酬を多くしすぎると同じプログラムを持った生物が多くなってしまうためにバリエーションが増えないためと考えられる。w_3（ペナルティ）に関しては，生物数，生物の種類とも顕著な違いを見つけることはできなかった。これは，差（3000，5000）が少なすぎたためと考えられる。

図7は，サイズ毎の個体数の変化を表したものである。紙面の都合上全系列は載せていないが，実行回数が進むと長さが先祖種（長さ80）よりも短いものが大半を占るようになった。これらの生物のプログラムを調べてみると，単独で自己複製が可能な生物は多くなかった。

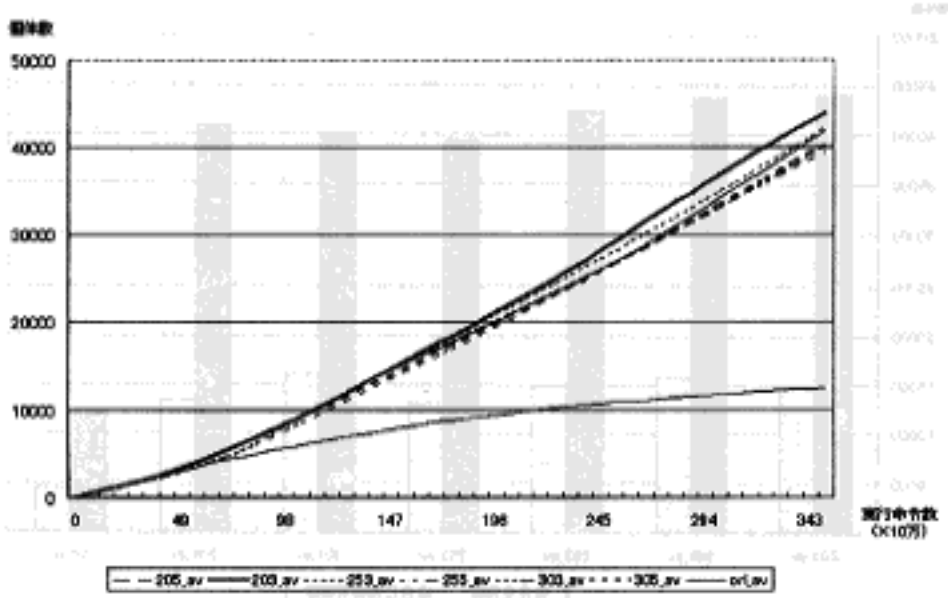

図 4　実行命令数における生物数の比較

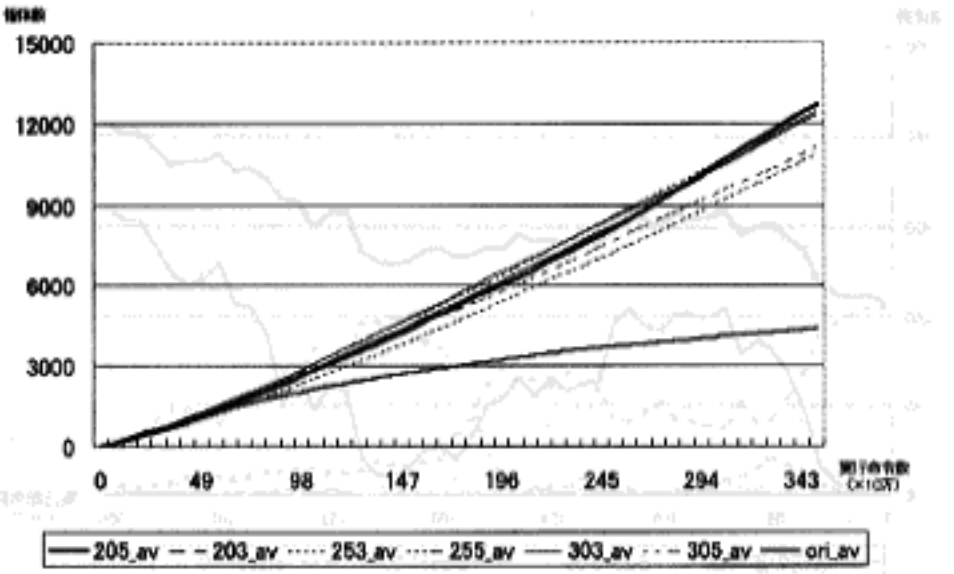

図 5　実行命令数における生物の種類の比較

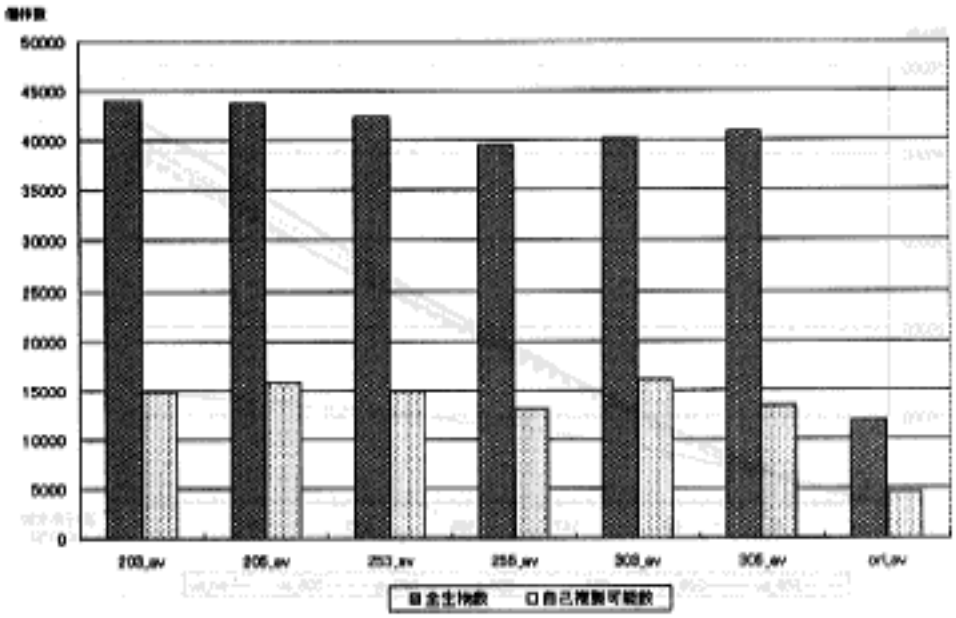

図6　3500万命令実行時における複製可能な生物数

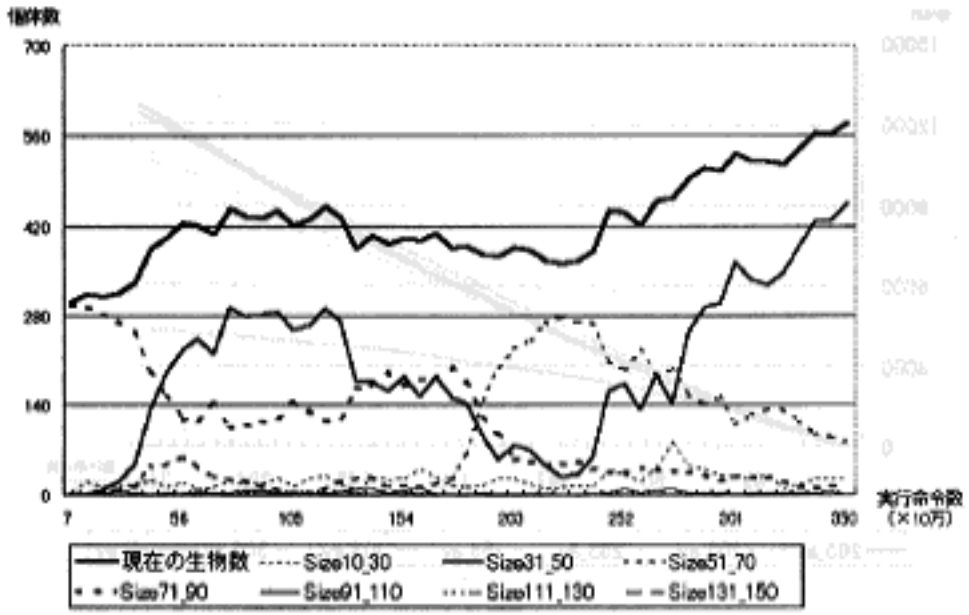

図7　生物のサイズの変化（系列205）

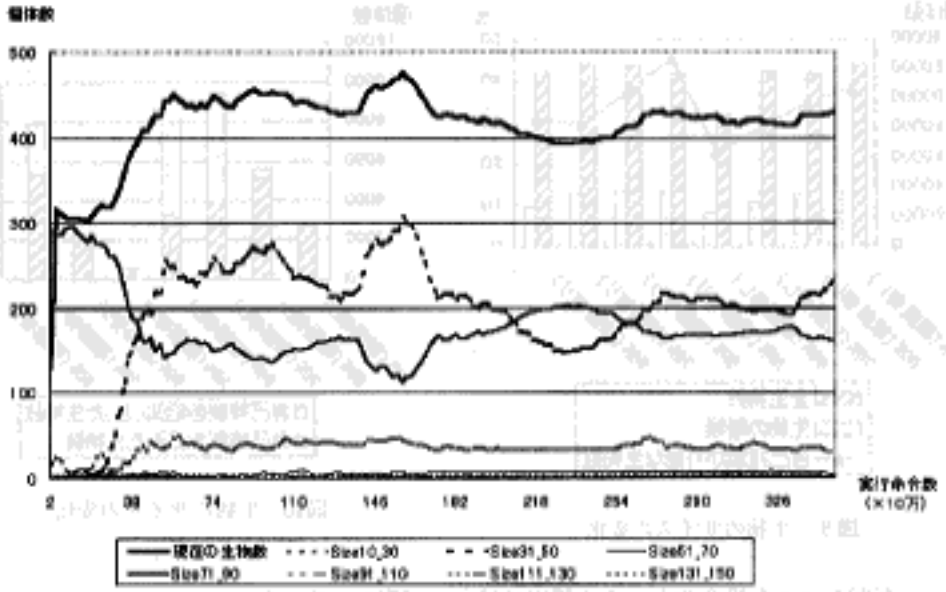

図8　生物のサイズの変化（オリジナル）

これは，実行できる命令の回数の制約（2，3，4）が少なすぎたのと，内部情報を子供に引き継ぐようにしたため，実行できる命令数が4回の生物が結果的にほとんどになってしまい，生物同士の格差が生まれなかったためと考えられる。

次に突然変異確率については，表4より自己複製を多くしている生物は，突然変異確率が低いことが確認できた。これによって，よいプログラムを持つ生物がそのままのプログラムでいる可能性が高くなっている。

表4　各系列における自己複製回数が多い生物

	生物1	生物2	生物3	生物4
系　　作	オリジナル	205	255	305
複製回数	4	7	7	11
突然変異確率	1	1/2	1/2	1/2
サ イ ズ	25	33	40	36
寿　　命	約43万	約70万	約87万	約130万

3.4　実験結果2

全ての場合において生物数，生物の種類，自己複製可能な生物数は，ω_1の比率を多くすると減少する(図9)。これは，自己複製可能な同一の生物が長く残りやすいので多様性が少なくなったためである。逆に一生物あたりの自己複製回数は増加している(図10)。また，移動プログラ

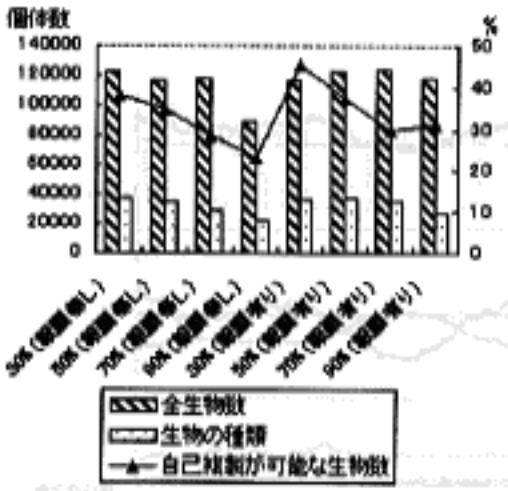

図9　生物のサイズの変化

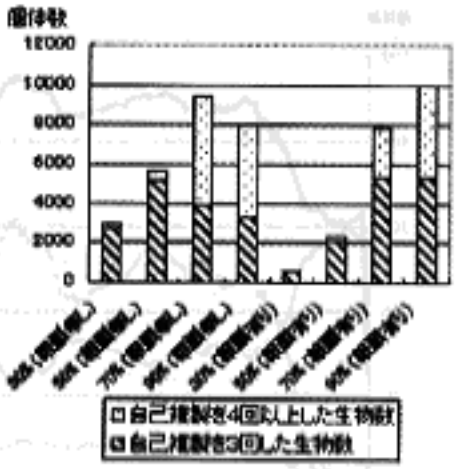

図10　生物のサイズの変化

ムから報酬を与えた場合でも，この傾向はほとんど変わっていない。

このことからωを変えることによって以下のどちらかの選択が可能と考えられる。

1．生物の種類が多く生まれるようにして，自己複製プログラムに対してより多くの変化を起こす（生物の種類重視）

2．一生物あたりの複製回数を多くして移動プログラムに対してより多くの変化を起こす（一生物の複製回数重視）

4．結　論

本報告では，効率のよいTierraの進化に関して以下の問題点を指摘した。

- リーパーによる淘汰に関する問題点
- CPU時間に関する問題点
- 突然変異確率に関する問題点

そして，次のような提案を行うことで問題点の解決を図った。

- 自己複製プログラムとは別のプログラムを持たせる多重プログラム型Tierraの基本構造を提案した。
- 複製回数による報酬及び命令の実行の失敗回数によるペナルティによって淘汰する生物を決定する方式を提案した。
  - スープ領域が残り20%を切ると，全生物に対して判別値を求めて淘汰を行う。
  - 複製に要する命令数に応じて報酬を与える。

－　並列プログラムの実行結果による一回に実行できる命令回数の変化が変化する。

－　並列プログラムの実行結果による突然変異確率が変化する。

今回の提案によって，

●オリジナルに比べ，より多くの生物及び種類が生まれることが確認できた。

●生物のサイズについて見てみると，ほとんどが寄生種と呼ばれる他の生物を利用する生物で，自分自身のプログラムで複製できるものではなかった。

●判別値におけるウェイトを変化させることによって，生物の種類重視か一生物の複製回数重視のどちらかの方針を選べることを示した。

これにより，適用する問題によってウェイトを変更して意図した淘汰を実現できると考えられる。また，本研究の発展としては，

●外部環境を導入しその環境変数を取り込み，生物の進化に影響を与える方向

●命令の実行回数に差をつけることによって生物のサイズに影響があるかを確認する

●生物に移動プログラムを持たせ，実際に２つのプログラムにどのような変化が起こるのかを観察する

などが考えられる。

参 考 文 献

[1] Ray, T.S：Evolution, cology and optimization of difital organisms, Santa Fe Institute working paper 92-08-042, 1992.

[2] 上田完次，下原勝憲，伊庭斉志：人口生命の方法，日本情報処理開発協会，1995.

[3] 木目沢司，Tierra 入門，WWW ホームページ，
http://www.hip.atr.o.jp/kim/TIERRA/tierr.html, 1996.

[4] 清水裕輔，菊地慶仁，桃内佳雄：遺伝情報を持つ Tierra システムの進化のメカニズムに関する研究，情報処理北海道シンポジウム 2000 講演論文集，pp 146-149，2000.

[5] 清水裕輔，菊地慶仁，桃内佳雄：多重プログラム型 Tierra に進化メカニズムに関する研究，精密工学会 2000 年度北海道支部会講演論文集，pp 88-89，2000.