平成 19 年度　　卒業論文

GNP の改良手法を用いたヘテロジニアスエージェントの学習

指導教員
舟橋健司　准教授
伊藤宏隆　助教

名古屋工業大学　工学部

情報工学科

平成 16 年度入学　16115007 番

池田直樹

目次

第 1 章

はじめに

引っ越しをしている時，一人でどうしても運べない重い家具があるとする．こんなときどうするか．家具を分割して運ぶことが出来ない場合，人を呼んで手伝ってもらう他ないであろう．

このようにある一個体で出来ないことを複数の力を足し合わせることによって行う作業が必要な場合が多く存在する．

近年，マルチエージェントシステムの研究が広く行われている．マルチエージェントシステムとはシングルエージェントでは解決できない問題を複数のエージェントを用いて解決するシステムである．個々のエージェントは自らが環境を知覚し目標を達成するように行動し，システム全体はあるエージェントの行動から環境が変化し，それによって別のエージェントの行動が決定される．マルチエージェントシステムではシステム全体をまとめて管理することはせず，個々のエージェントが協調することによってシステムを成り立たせる．このように分散して管理することでシステム全体のコストが低くなる利点がある．

ここでエージェントを知覚することと動作することに分け，知覚することは環境を感知する能力，動作することは環境に影響を与える能力と考える(図 1.1)．このときエージェントにタスクを与えたときの行動は，それが持つ環境を感知する能力と環境に影響を与える能力をどのような手順で扱うかで決定される．

図 1.1：エージェントの能力

タスクを達成することとは環境を感知した後に環境に影響を与えることを行い，さらにそれらの行動後に他の行動を行ってまた環境を感知させる，ということを繰り返して行うことである．単純なタスクを達成するエージェントを作成することは手作業でその行動を設計することが可能であるが，複雑なタスクの場合多くの時間と労力がかかることがある．さらに，複数のエージェントをそのタスクに対する役割に応じて行動の設計を行うことはより複雑な問題となるので手作業で行動を設定することは難しくなる．このようなタスクの設定を手作業で行うことの難しさは環境を感知する行動と環境に影響を与える行動をどのように組み合わせて用いるかということが複雑になることで起こる．

このような問題に対してエージェントの行動を自動生成する手法として遺伝的ネットワークプログラミング(Genetic Network Programming : GNP)[1]が存在する．GNP は遺伝的アルゴリズムと遺伝的プログラミングを元に考えられた手法で，生命の進化を利用して環境に適応出来るエージェントを組み合わせ最適化問題を解くことにより生成することが可能である．GNP では有向グラフ構造を持つことで，エージェントの行動により環境が変化する動的な環境に対応することが可能なエージェントを作成することが出来る[2][3][4].

本文では動的な環境で従来手法の GNP とその改善手法である免疫型進化手法を用いた GNP(Immune evolved GNP : IGNP)[5]，免疫調節機構を持つ GNP(GNP with Immune Adjustment Mechanism : GNPIAM)[6]を用いて複数のエージェントの共同作業を行うタスクにおいて改善手法の有効性について検討を行う．共同作業とは単体のエージェントでは達成出来ないタスクとし，複数のエージェントでのみ達成できるタスクとする．従来の GNP を用いたマルチエージェントシステムでは，シングルエージェントで達成出来るタスクに対して複数のエージェントを用いることで効率を上げるものが主であったが[7][8][9]，本研究では複数のエージェントで効率を上げるのではなく複数のエージェントが共同することで達成するタスクに対応するエージェントの作成を行う．つまり，小さい能力のエージェントを複数用いることで大きい能力のエージェントと同等の能力を発揮することが可能なシステムを構築することを目標としている．

また，ヘテロジニアスエージェントの進化方法として共進化手法と非共進化手法を提案し，どちらの手法の性能が高いかも評価した．共進化手法とは各エージェントにタスク達成への貢献した割合に応じて評価値を与え，それによって各エージェントが個別に進化する手法である．非共進化手法は複数のエージェント全体に評価値が与えられ，それに応じて全体が進化する手法である．

以下，2 章で本研究に関する基礎知識を述べ，3 章で GNP を用いたマルチエージェントシステムのモデルを示し，4 章でタイルワールドを用いた実験の方法と環境，結果を述べたあと，5 章で研究全体を通しての考察と今後の課題について述べる．

第二章

背景知識

2.1　遺伝的アルゴリズム(Genetic Algorithm : GA)

GA は生物の進化における遺伝のメカニズムを利用した最適化アルゴリズムである. GA の解は 1 行の数列からなる遺伝子を持つ個体として表現され, 複数体用意される(図 2.1). 設定した問題に適した遺伝子を持つ個体は多く子孫の残すことができ, 適さない遺伝子を持つ個体は子孫をほとんど残さずに死滅していく. 子孫は 2 体の個体の遺伝子を交叉, 突然変異することで生み出される. GA の進化の流れを図 2.2 に示す.

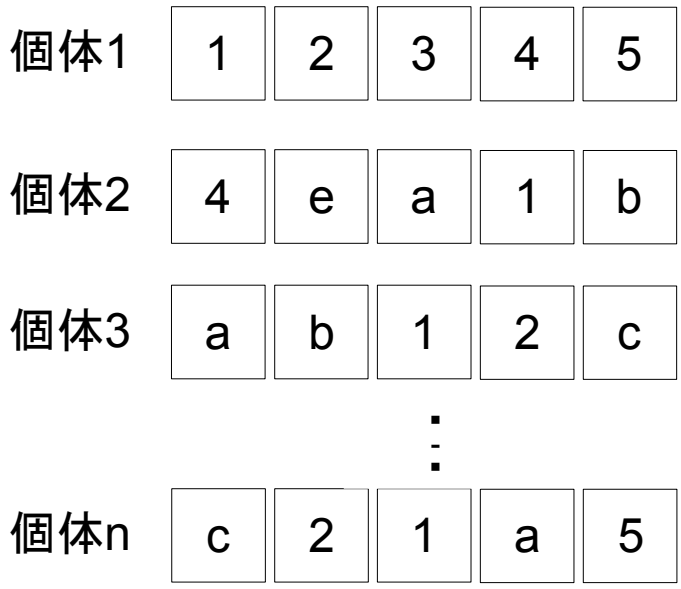


図 2.1 : 個体の遺伝子

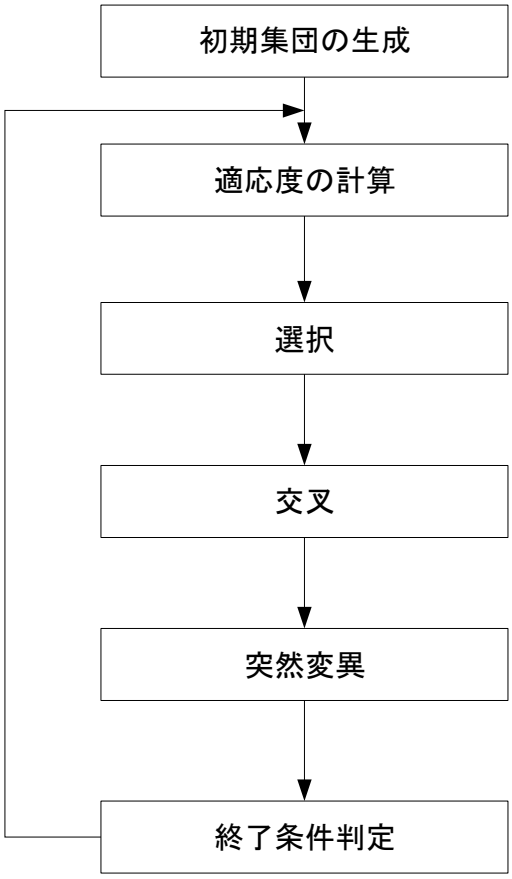


図 2.2 : 遺伝アルゴリズムの流れ

GA は以下のプロセスで行われる.

(1) 初期集団の生成

初期世代の個体をランダムに生成する.

(2) 適合度の計算

問題に対して個体がどの程度解けているかの適合度の計算を行う.

(3) 交叉

適合度を参考にして 2 個体間で遺伝子の一部分を交換させる操作を行う(図 2.3).

(4) 突然変異

個体の遺伝子の一部を別のものと置き換える操作を行う(図 2.4).

(5) 終了条件判定

決められた世代数に達するまで(2)から(4)を繰り返す.

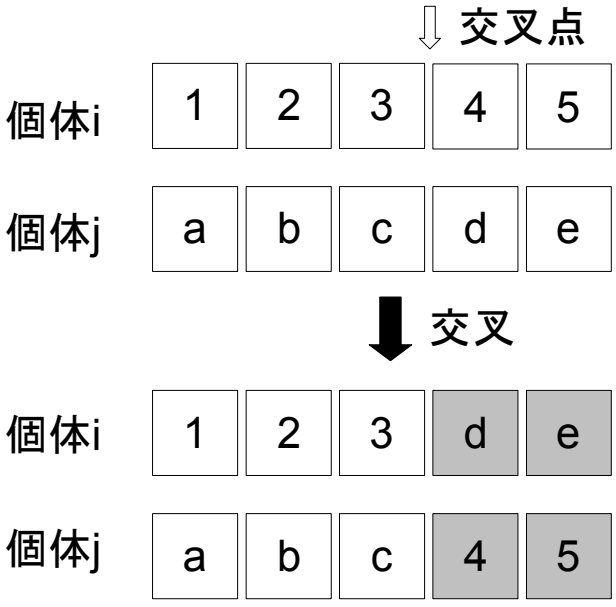


図 2.3: 交叉操作

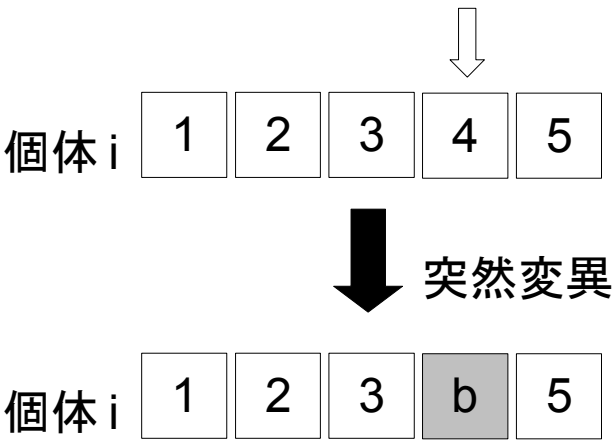


図 2.4 : 突然変異操作

GA は有効な解のアルゴリズムが見つからないとき構造の単純さから広く用いられる. しかし, 初期収束という初期の段階で適合度の高い個体が生まれるとそれ以降適合度が上がらなくなり, 広い探索空間を探索できないという問題がある.

2.2 遺伝的ネットワークプログラミング(Genetic Network Programming : GNP)

遺伝的ネットワークプログラミングは GA，進化的プログラミング(Evolutionary Programming:EP)，遺伝的プログラミング(Genetic Programming:GP)を元にした自動プログラミング手法の一つである．

EP と GNP は両方とも有向グラフ構造を持つが，EP はオートマトンとして有向グラフを扱うため全入力に対する遷移を割り当てるのに対して，GNP はノードを種類に分けることで必要な分だけの遷移を割り当てることによって EP よりも単純な構造になっている．

また，GP は木構造を持ちエージェントの行動規則を学習することができるが，終端に遷移した後ルートに戻ることでループする構造から，主に静的な環境に対しての学習に向いている．それに対し GNP は有向グラフ構造によってエージェントの行動規則を学習することが可能で，行動の構造を部分グラフ内ループで行うので動的な環境に対して学習することに向いている．

GNP の遺伝子はネットワーク上のノード集合として扱われ，ノードには開始ノード，処理ノードと判定ノードがある．開始ノードは処理の開始地点であり，処理ノードはエージェントの動作部として扱われ，判定ノードはセンサー部として扱われる(図 2.5)．

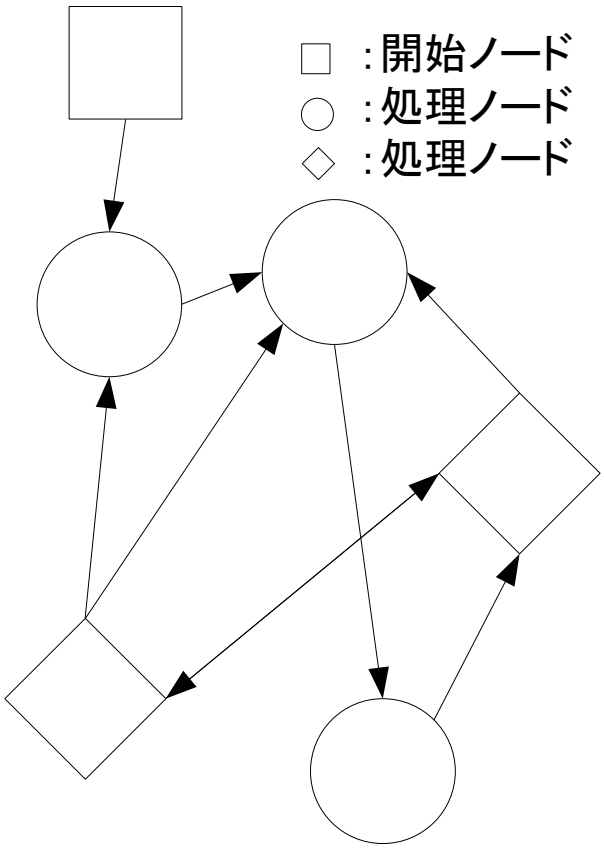


図 2.5: GNP のデータ構造

ノードはノードの種類，ノード ID，遅れ時間をあらわす部分と他のノードへのリンク，リンクの遅れ時間を表す部分で構成される．ノードはノードライブラリにあらかじめ登録される(図 2.6)．

GNP の動作はまず開始ノードからはじめ，他のノードにリンクをたどって遷移した後スタートノードには戻らずにノード間を遷移する．また，GNP には実行の制限時間が設けられ，ノードを通過する際にはノードの遅れ時間を制限時間から減らしていくことで，ある限られた時間内に何らかの処理を実行する事を保証している．また，GNP はあるノードの前に別のノードを通過しているので，過去の情報を構造的に記憶するメモリのようなものを持っていることになる．

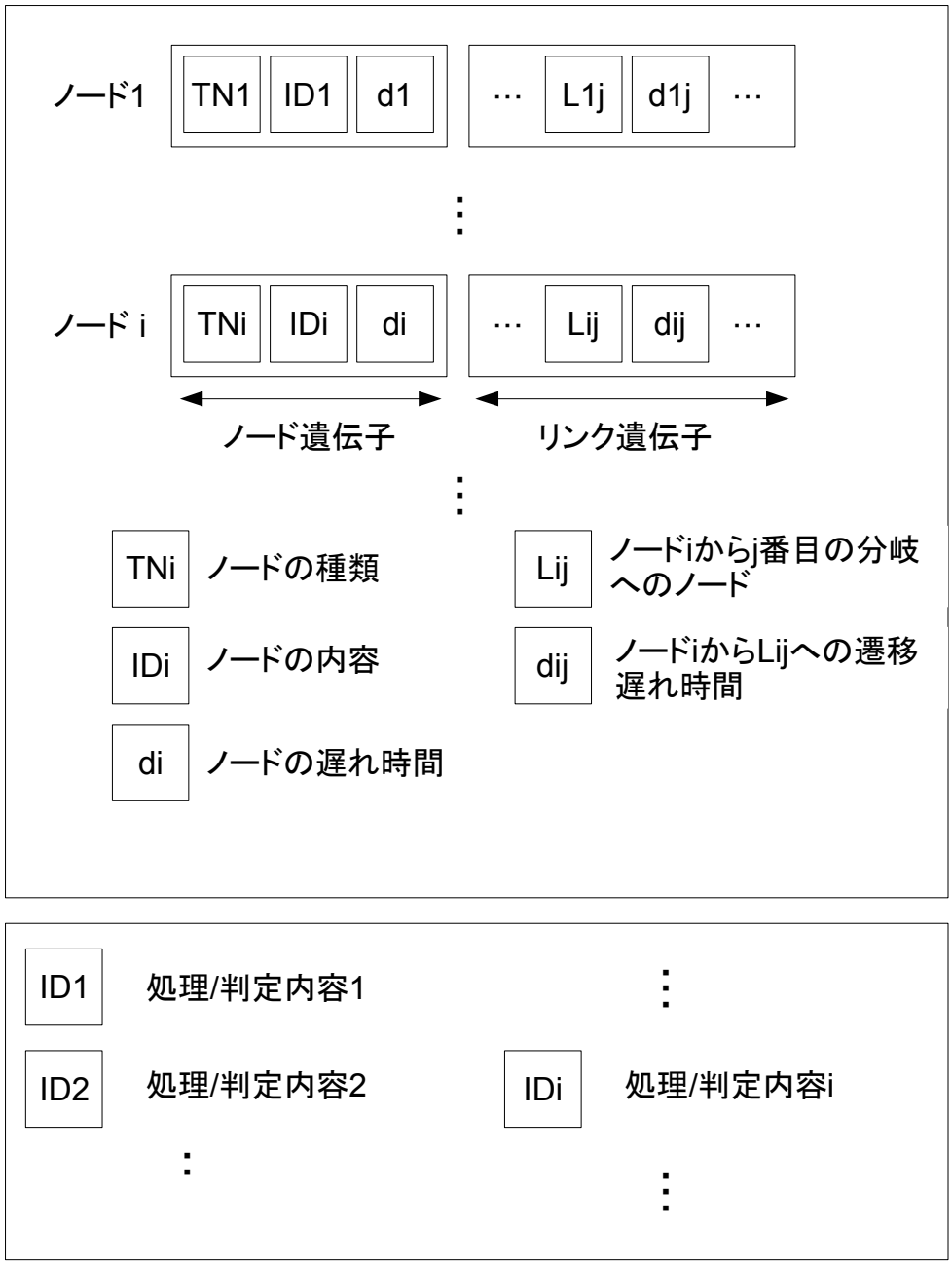


図 2.6 : ノードライブラリ

進化の流れは GA と同じであるが, GNP は有向グラフ構造なので GA とは遺伝的操作である交叉, 突然変異の方法が異なる. GNP の交叉は図 2.7 のように特定の領域の入れ替えによりリンクの張替え, ノードの付け替えが行われる. 突然変異はリンク突然変異とノード突然変異があり図 2.8, 図 2.9 のように特定のノード, リンクに対して張替えと付け替えが行われる.

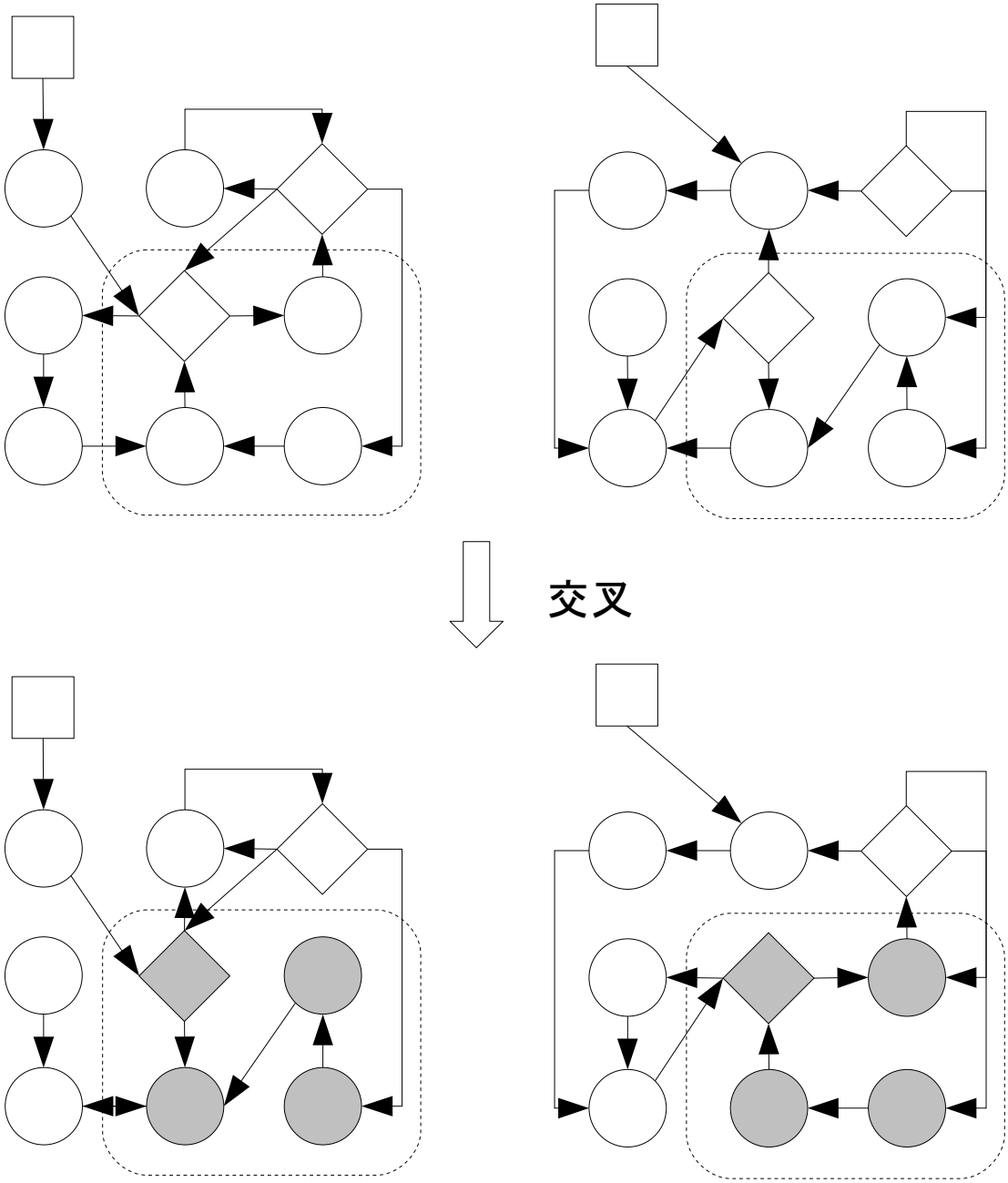
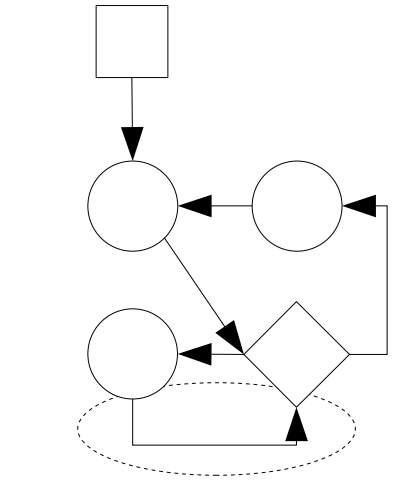


図 2.7 : GNP の交叉操作



リンク突然変異

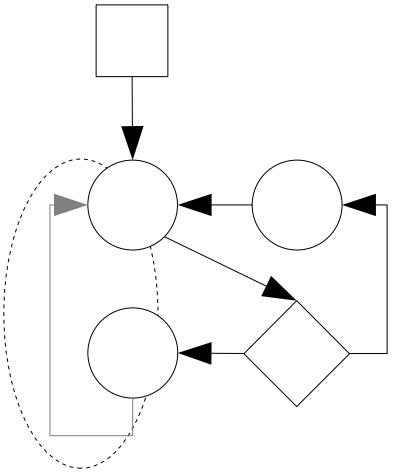
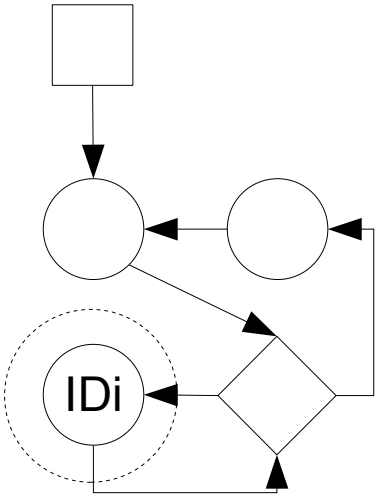


図 2.8 : GNP のリンク突然変異



ノード突然変異

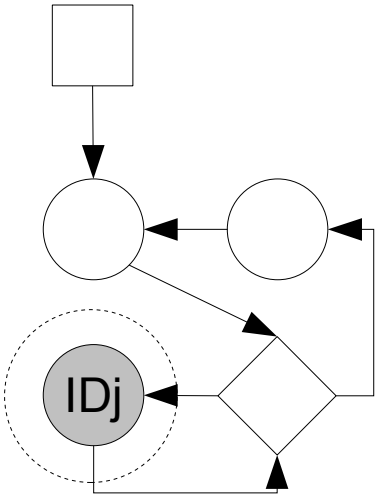


図 2.9 : GNP のノード突然変異

本研究の GNP は突然変異にリンク突然変異のみを採用した. これにより遺伝子構造が GA と同じものとなり GA と同じ遺伝的操作で進化計算を行える.

2.3　免疫アルゴリズム(Immune Algorithm : IA)

免疫アルゴリズムは脊椎動物が持つ獲得免疫機構を利用した, GA の初期収束の問題を解決したアルゴリズムである. 初期収束とは初期の段階で適合度の高い個体が生まれてしまったとき, その個体に類似した個体が増えてしまうことで局所解に陥ることである.

免疫アルゴリズムは問題を抗原として扱い, 解を抗体として扱う. 抗体が抗原に対してどの程度対応できるかを抗原との親和度, 他の抗体とどの程度類似しているかを抗体との親和度とする. 免疫機構は抗原が現れたときその抗原に対応する抗体を記憶し, 抗原に抗体が作用した後は抗原との親和度の高い抗体を減らす抑制を行う.

以下のプロセスで行われる(図 2.10).

(1) 初期集団の生成

初期の抗体群をランダムに生成する.

(2) 抗原との親和度の計算

抗原との親和度 opt_v から正規化して 0 から 1 までの値を評価値 fit_v とする.

$$fit_v = \frac{1}{1+opt_v}$$

(3) 抗原との親和度の高い抗体を記憶

評価値が高い抗体を記憶細胞に追加する.

(4) 記憶細胞による抑制

記憶細胞に入っている抗体と親和度が閾値以上高い抗体を削除する.

(5) 濃度の計算

全抗体に対して親和度を計算し, 濃度を求める. 抗体 v に対する濃度 C_v は以下のように計算する.

$$C_v = \sum_{i=1}^{N} g_{vi}$$

g_{vi} は i 番目の抗体との親和度がある閾値以上のとき 1, その他は 0 とする. C_v は自分との親和度を求めているため常に非 0 である.

(6) 濃度による抑制

濃度と評価値，記憶細胞との親和度から期待値を求め，期待値が低い抗体を削除する．期待値 e_v は以下のように計算する．

$$e_v = \frac{fit_v}{C_v}$$

このように評価値が高い且つ濃度が低いときに期待値は高くなるように設定する．

(7) 遺伝的操作

(6)で削除した抗体に代わる抗体を遺伝的操作のうち交叉，突然変異によって産生する．

(8) (2)から(7)を繰り返す

IA は解となる抗体同士が類似しているかどうかを評価し，削除して新しい抗体を作ることによって探索空間を広げ GA の初期収束の問題を解決することが可能となる．

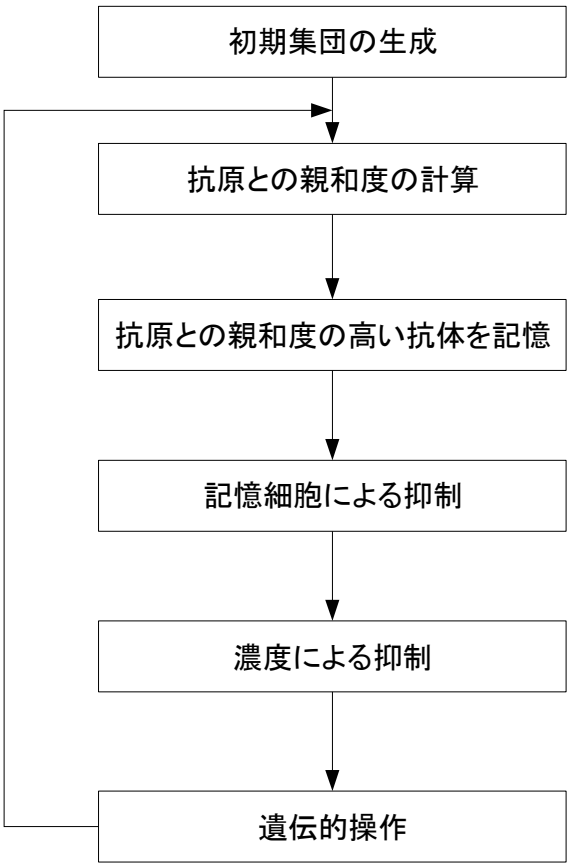


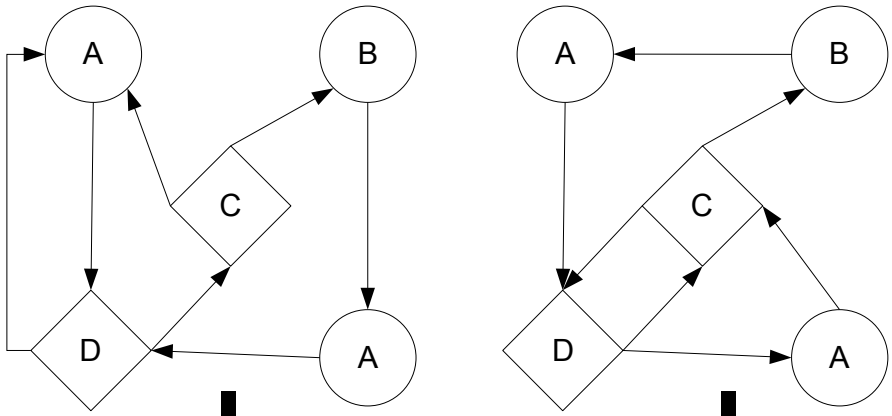
図 2.10 : IA の流れ

2.4 免疫型進化手法を用いた遺伝的ネットワークプログラミング (Immune evolved GNP : IGNP)

IGNP は初期収束を解決するために GNP に免疫アルゴリズムを適用したアルゴリズムである. 進化のプロセスは IA と同じであるので, ここでは抗体間の親和度, どの程度抗体間で似ているかの類似度の算出方法のみ述べる.

GNP での有向グラフデータについての類似度は, あるノードからあるノードへのリンクが張られていることから求められる. 文献[5]での類似度の算出方法と, 本文で用いた類似度の算出方法の違いを述べる.

文献[5]での類似度の算出方法は図 2.11 のようになる.



	A	B	C	D
A	0	0	0	2
B	1	0	0	0
C	1	1	0	0
D	1	0	1	0

	A	B	C	D
A	0	0	1	1
B	1	0	0	0
C	0	1	0	1
D	1	0	1	0

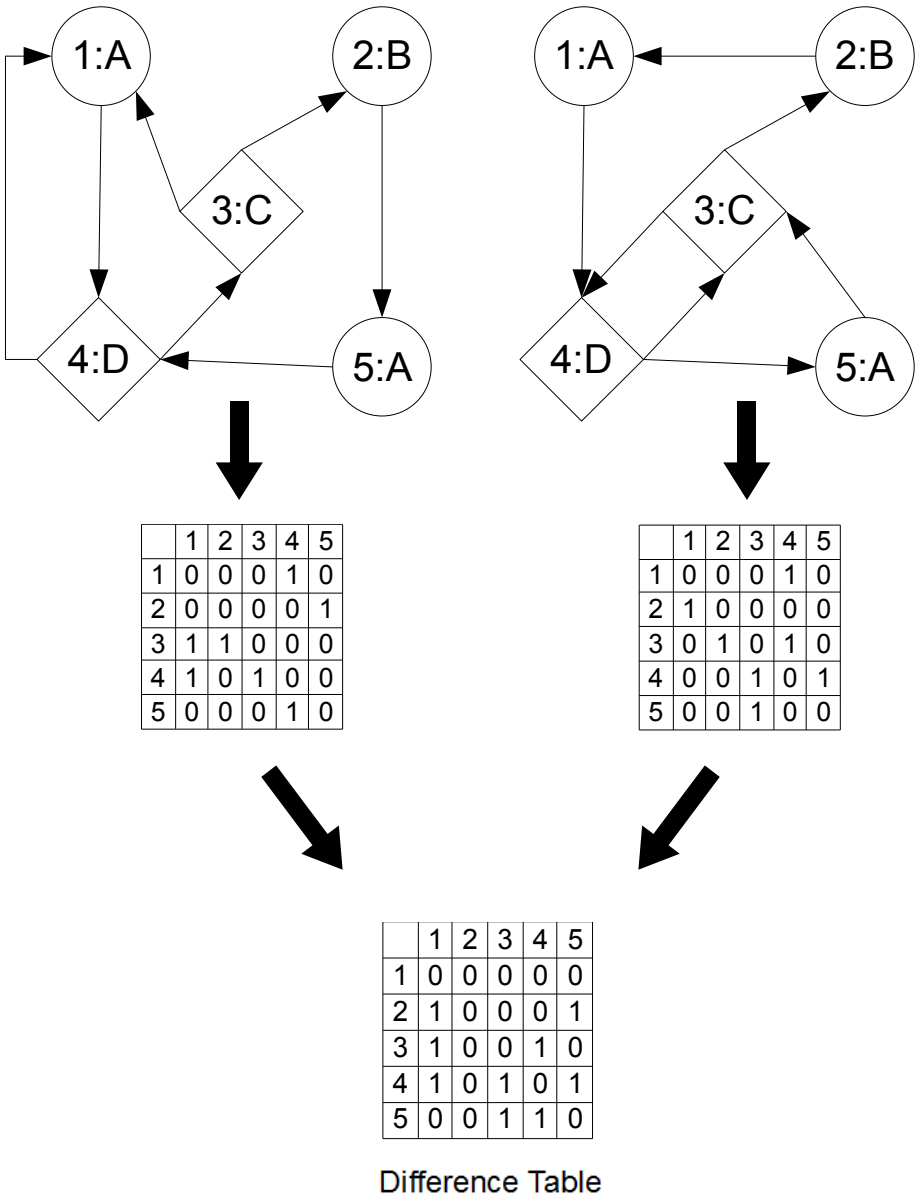
	A	B	C	D
A	0	0	1	1
B	0	0	0	0
C	1	0	0	1
D	0	0	0	0

Difference Table

図 2.11 : 文献[5] での類似度の算出方法

A, B, C, D はそれぞれノードの種類を表す. 類似度の算出はノードの種類についてどの種類のノードへのリンクがあるかどうかによってテーブルを作成することで行う. 各有向グラフについてテーブルを作成し, 2つのテーブルから各要素の差分の絶対値を取り新しいテーブルを作成する. このテーブルを用いて, ノード毎の類似度を算出する. ノード毎の類似度はその種類のノードの数とノードから伸びる分岐数から 0 から 1 に正規化される.

本稿での類似度の算出方法は図 2.12 のようになる.



	1	2	3	4	5
1	0	0	0	1	0
2	0	0	0	0	1
3	1	1	0	0	0
4	1	0	1	0	0
5	0	0	0	1	0

	1	2	3	4	5
1	0	0	0	1	0
2	1	0	0	0	0
3	0	1	0	1	0
4	0	0	1	0	1
5	0	0	1	0	0

	1	2	3	4	5
1	0	0	0	0	0
2	1	0	0	0	1
3	1	0	0	1	0
4	1	0	1	0	1
5	0	0	1	1	0

図 2.12 : 本稿で用いた類似度の算出方法

有向グラフ上に配置されたノードに対して固有のインデックスを付け，各ノードについてのリンク情報をテーブルとする．2 つのテーブルから各要素の排他的論理和を取ることによって新しいテーブルを算出する．新しいテーブルの全ての要素の和を用いて 0 から 1 に正規化し類似度とする．

2 つの算出方法の違いは，有向グラフから作成したテーブルから元の有向グラフが作成できるかどうかである．文献[5]の算出方法では，作成したテーブルから元の有向グラフには戻すことが出来ない．つまり，作成したテーブルは元の有向グラフの性質をすべてあらわしているとは言えない．本稿での算出方法では，ノードの数が多くなった場合にテーブルのサイズが巨大になるため効率が悪くなるが，類似度の算出方法としては単純なためこの方法を採用した．

2.5　免疫調整機構による遺伝的ネットワークプログラミング(GNP with Immune Adjustment Mechanism : GNPIAM)

GNPIAM は IGNP[5]と GAIAM[6]を参考にして作成した GNP の改良手法である. GAIAM は IA を改良したアルゴリズムであり, 局所探索の抗体数を下げる代わりに大域的な探索を行えるようにしたものであり, 局所探索を行う個体を分別し無駄な進化を防ぐことで解の向上が行われている. 免疫アルゴリズムと同様解を抗体として扱い, 抗原を問題として抗原との親和度は評価値を表す.

プロセスは以下のようになる(図 2.13).

(1) 初期集団の生成

初期の抗体群を N 個ランダムに生成する.

(2) 抗原との親和度の計算

抗原との親和度を評価値として計算する.

(3) 期待値の計算

抗体間の親和度を求め, 親和度の和と評価値から期待値を求める. 求めた期待値の大きさにしたがって $\frac{N}{2}$ 個の抗体を消滅させる. ただし評価値の上位 10%は消滅の対象外とする.

(4) 抗体産生

(3)で消滅させた $\frac{N}{2}$ 個に代わる新しい抗体を残った $\frac{N}{2}$ 個から期待値の大きさにしたがって選択し, それを突然変異させて $\frac{N}{2}$ 個の抗体を産生し, 全抗体を N 個に戻す.

(5) 調節用抗体の生成

N 個の抗体から重複を許してランダムに抗体を選び, 交叉と突然変異を確率に応じて行い調節用の抗体として $\frac{N}{2}$ 個産生し, 評価値を計算する.

(6) 抗体間の調節

(5)で新しく作られた $\frac{N}{2}$ 個の各調節用抗体 i に対して， N 個の既存の抗体と最も類似度の高い抗体 j を探す. 抗体 i と j の評価値を比べ高い方を次世代に残す. 類似度の算出方法は IGNP と同じものを用いる.

(7) (3)から(6)を繰り返す

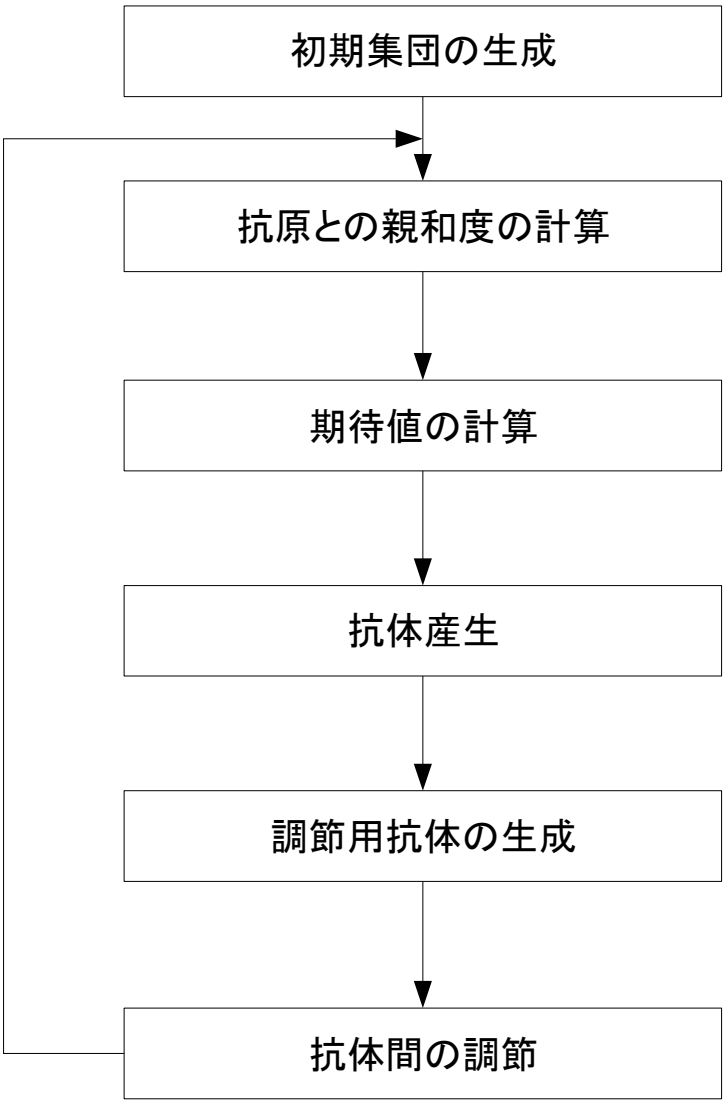


図 2.13 : GNPIAM の流れ

第三章

GNP によるマルチエージェントシステム

3.1　マルチエージェントシステム(Multi Agent System : MAS)

MAS とは個々が自立的に動くエージェントを利用し，複数のエージェントで全体のシステムを構築するものである．エージェント同士が協調することで単一のエージェントではこなせない大きなタスクをこなせる可能性がある．エージェントは自分の行動プログラムを示す内部と，環境に影響を及ぼす外部という2つの部位で成り立っている．

エージェントは内部の行動プログラムを変更することが出来ても，外部に影響を与える大きさを変更することは困難である場合が多い．また，条件として外部に及ぼす影響が一定以上大きなエージェントを作成できない場合，外部に対して小さな影響を与えるエージェントのみを用いるしかない場合が存在する．このようなことから環境に対して小さい影響を及ぼすエージェントを多数用意することで，環境に対して大きい影響を及ぼすことが出来ることが求められる．

このような問題をマルチエージェント問題といい，ある種のタスクを MAS の強調で効率的に解決するものであり，また，そのように動作するエージェントを効率的に作成することが求められる．

3.2　GNP を用いた MAS の構成

MAS にはヘテロジニアスエージェントとホモジニアスエージェントというエージェントの違いによって構成の違いが生じる．システムを構築するために集められたエージェントをエージェント群とする．GNP で扱うエージェントは，内部プログラムつまり有向グラフがエージェント群内で違うものをヘテロジニアスエージェント，有向グラフが同じものをホモジニアスエージェントとし，外部に与える影響は同じ大きさであるものとする．GNP で可能なマルチエージェントを以下述べていく．

3.2.1 1つの有向グラフを用いるホモジニアスエージェント

一つの有向グラフを学習し，エージェント群が同じ有向グラフを用い，このようなエージェントをホモジニアスエージェントと言う(図 3.1).

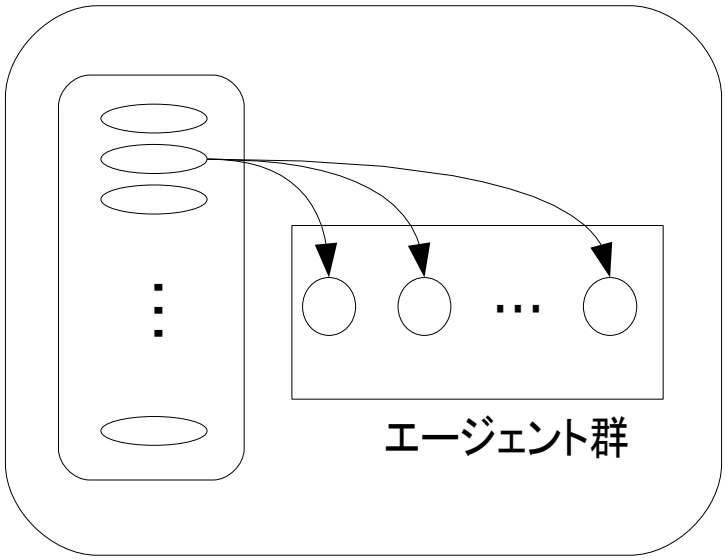


図 3.1：ホモジニアスエージェント

複数のエージェントが同じ環境においてタスクを行い，環境の変化が生じない場合は有向グラフ内の遷移はすべて同じとなり全てのエージェントが同じ行動を行う．荷物を持つ人の例とすると，複数人がある荷物を持とうとするときに持ち上げる動作を全く同じに行うため，何も掛け声をかけずに荷物を運ぶことが可能である．

3.2.2 1つの有向グラフ内に複数の開始ノードを持つエージェント

GNP には並列動作という開始ノードを複数おくことで並列に動作することが可能となり，ホモジニアスとヘテロジニアスの中間に当たるエージェントを構成することが出来る(図 3.2).

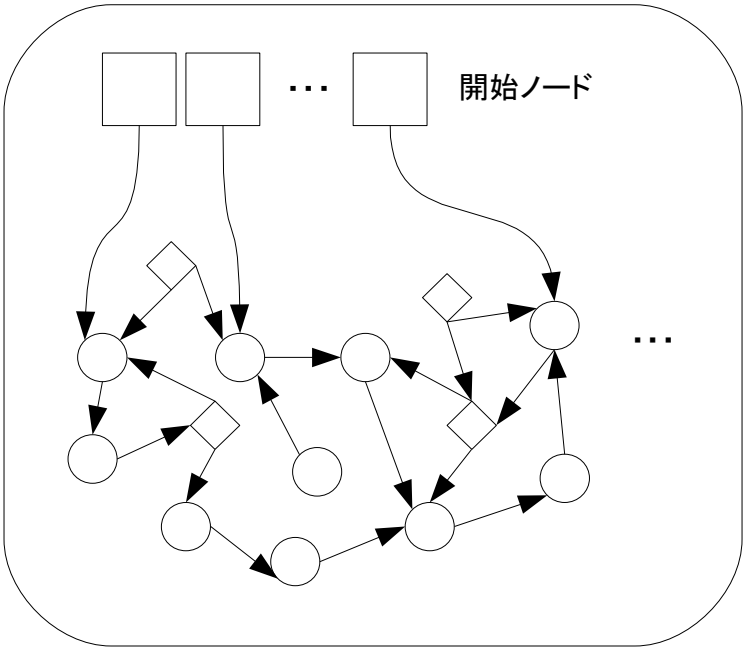


図 3.2：複数の開始ノードを持つ有向グラフ

しかし，複数のエージェントの開始ノードからのリンクが有向グラフ内の近い位置にあるとき，エージェントは同一動作をするので 3.2.1 で述べたエージェントと動作については変わらない．また有向グラフの学習が進むことを考えると，複数のエージェントの開始ノードがある一部分に集中することは容易に考えることができる．このことは，部分有向グラフの学習が進むことで全体の評価値が向上するためである．よって，実質はホモジニアスマルチエージェントになってしまうと考えられる．このことから本研究では 3.2.1，3.2.2 のような MAS は扱わない．

3.2.3 非共進化型ヘテロジニアスエージェント

非共進化型ヘテロジニアスエージェントとは，個々のエージェントが個々に有向グラフを持つがエージェント同士間に評価値による区別はないものである．一つの有向グラフに複数の開始ノードを用意し，区切ることによって複数のエージェントが使用する．エージェント間の区別とはシステム全体で見た場合の区別で，あるタスクをこなした場合にそのタスクへの貢献度をエージェントで区別するかどうかである．このとき区切った部分間ではリンクが張られないようにし，独立した部分有向グラフとする(図 3.3).

このように構成したヘテロジニアスマルチエージェントは，部分有向グラフが複数ある有向グラフと見ることが出来るので，部分グラフ同士にリンクを張らないようにすることで通常の GNP と同じように進化処理を行うことが可能である．

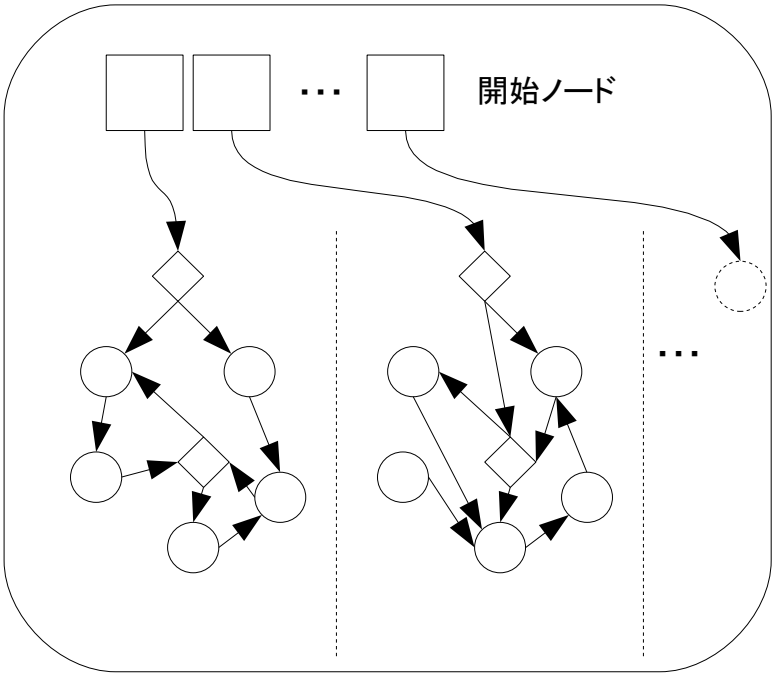


図 3.3：有向グラフ内を分割

3.2.4 共進化型ヘテロジニアスエージェント

共進化型ヘテロジニアスエージェントとは，個々のエージェントが個々に有向グラフを持ち，さらにそのエージェント同士間にタスクへの貢献度の区別があるものである．つまり，個々のエージェントに対して評価値を与えることで，エージェントを個別に進化させるものである．ここでの共進化とはエージェント群内での共進化という意味であり，あるエージェントの学習は別のエージェントの学習に影響を与えるということである．このようにエージェント群内でタスクに貢献したエージェントとそうでないエージェントの区別をつけることで，タスクに貢献していないエージェントを貢献できるように学習できると考えられる．

このようにする理由はあるエージェントが適合度のすべてを支配してしまうことが起こった場合，他のエージェントの進化を妨げ全体として進化出来ないことが起こりうるためである．これは，進化が適合度の高い個体が多く残っていくことから生じるものであり，適合度の順番がある特定のエージェントの得ることが出来た適合度で決定されるという意味である．これによってあるエージェントは進化が進むが，他のエージェントは進化が進みづらいという欠点がある(図 3.4)．

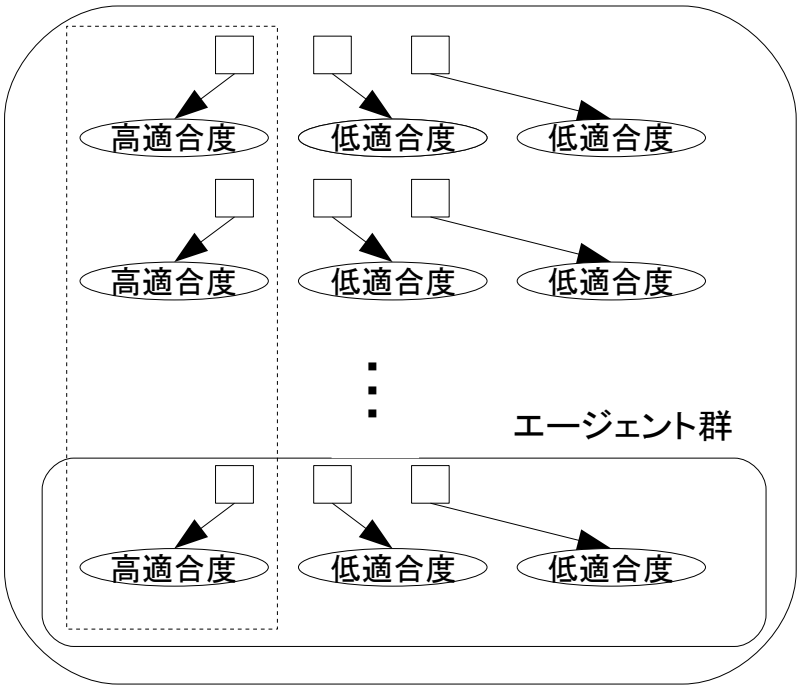


図 3.4：順番があるエージェントに大きく影響する場合

これを回避するため，タスクを行ったときに得られる評価値をそれぞれのエージェントが行った量に対してそれぞれ与え，それぞれのエージェントごとに進化処理を行う．エージェントごとに進化を行うことで，他のエージェントの行動に影響され共進化することで自分の行動が学習される．

進化計算を以下のように行うことで共進化を行う.

(1) 初期集団の生成

初期集団ランダムに作成する.

(2) 初期集団からエージェント群の作成

用いるエージェント数をエージェントグループ数としてグループごとにまとめ, エージェントグループから各 1 個体ずつを選択してエージェント群とする(図 3.5). このエージェント群は進化処理によって変更されない.

(3) タスクを行う

エージェント群で, タスクを行う.

(4) エージェントごとに適合度を求める

タスクに対する達成度から個々に適合度を求める.

(5) エージェントごとに進化処理を行う

エージェントグループごとに進化処理を行う.

(6) (3)から(5)を終了条件に来るまで繰り返す

共進化はグループ間の進化とも考えることができ, グループごとに異なった方向へ学習が進み, タスクに対する分業が行われることも期待してこのようにした.

本研究では, 3.2.3 と 3.2.4 のヘテロジニアスマルチエージェントを用いて, 動的な環境の共同作業の可能なシステムの検討を行う.

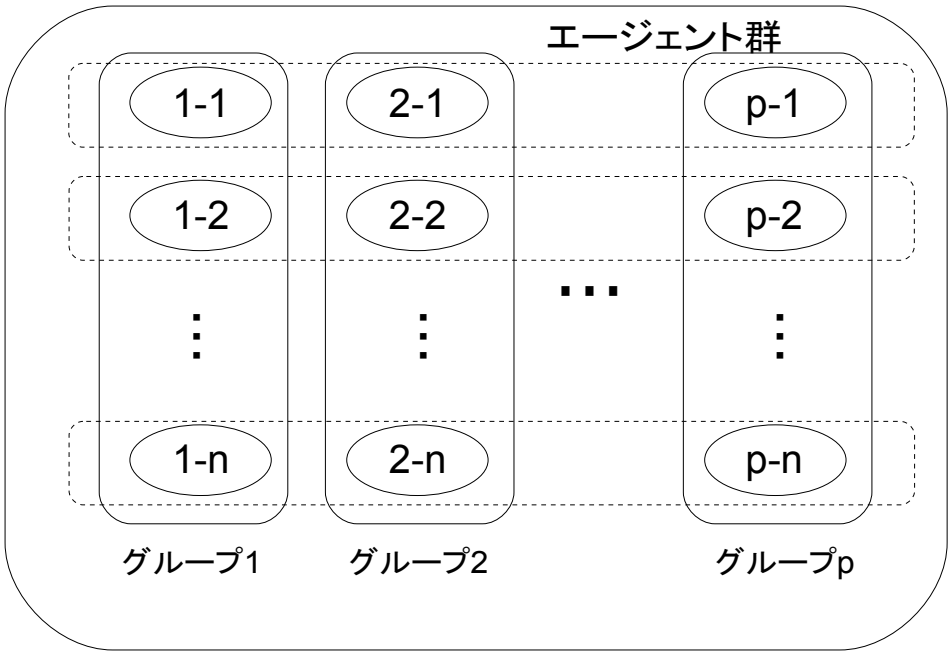


図 3.5 : エージェント群内の各エージェントのグループ分け

3.3 GNP による同期について

複数の GNP によって作成したエージェントが存在するとき，あるノードをある時間に実行する必要がある場合がある．本研究でエージェントが行うタスクで重要となるのは，エージェント同士が同期を取ることである．ここでいう同期とは，GNP内のある特定の種類のノードをある時間に同時に行うことを示し，同期を取ることで初めて共同作業の可能なエージェントとなる．

図 3.6 のような GNP のマルチエージェントシステムの時間の流れから同期の例を示す．

時刻	0	1	2	3	4	5	6	7	8
エージェント1	処理2		処理3			処理1	処理2		処理1
エージェント2	処理3			処理2		処理1	処理3		

図 3.6 : GNP のマルチエージェントシステムの時間の流れの例

ここで処理 1 にかかる時間を 1，処理 2 にかかる時間を 2，処理 3 にかかる時間を 3 とし，処理 1 はエージェント 1 とエージェント 2 が同時刻に行わないと意味が無いとする．

時間 5 の時点で処理 1 がエージェント 1,2 共に実行され，意味のある行動が可能となるが，時間 8 ではエージェント 1 のみが処理 1 を実行しているので意味のある行動とはならない．時間 5 で起こっていることは，たとえば重い物を 2 人で運ぶとき，両者が同時に持ち上げることを意味する．また，時間 8 で起こっていることは重いものを二人で運ぶときに，片方の人が持ち上げようとしているのにもう片方の人はほかの事をしていることを意味する．

第四章

実験と考察

4.1 シミュレーション環境

動的な環境をシミュレートする例としてタイルワールドにおけるタイル運びというタスクがよく知られている. タイルワールドとは図 4.1 に示すようなタイル, 床, 穴, 障害物が配置された 2 次元格子平面である. 格子内の一つ一つの区画をセルと呼び, エージェントは単位時間に 1 セル動くことが可能である. エージェントによってこのタイルワールドでタイルを穴に落とすことの評価を行う. エージェント同士, エージェントとタイルは同じセルを占めることができるが, タイル同士は同じセルを占めることが出来ないものとする.

	1	2	3	4	5	6	7	8	9	10	11	12
1	■	■	■	■	■	■	■	■	■	■	■	■
2	■											■
3	■			T								■
4	■					○			T			■
5	■											■
6	■											■
7	■				T							■
8	■							A				■
9	■											■
10	■		○									■
11	■											■
12	■	■	■	■	■	■	■	■	■	■	■	■

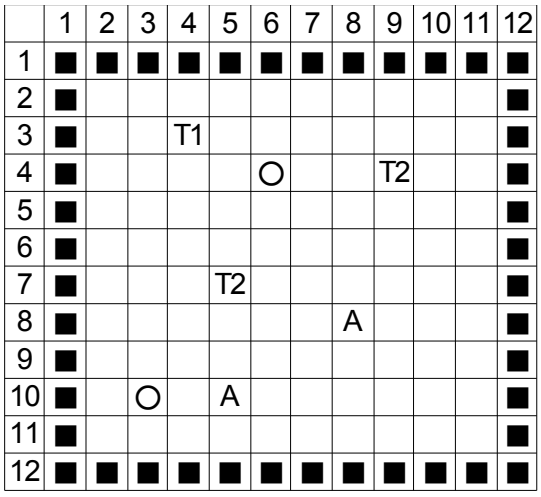
図 4.1 : 10×10 のタイルワールド

T はタイル, A はエージェント, ○は穴, ■は障害物を意味する

4.2 複数のエージェントでのタイルワールド

本研究ではこのタイルワールドを拡張し，タイルごとに重さを加える．重さとはタイルを運べる最少エージェント数を表し，重さが n のとき最低 n 体のエージェントが協調して運ぶ必要があることを意味する(図 4.2).

重さ 1 のタイルを穴に落とす動作は，1 体以上のエージェントがまずタイルのあるセルに入り，タイルを掴んだ後タイルを穴のある方へ移動させ，穴のあるセルに入ったあとタイルを放すことで行われる．重さ n のタイルを移動する動作は，n 体以上のエージェントが 1 つのセルに集まり，タイルを掴んだ後同じ方向に対して移動することで行われる．重さ n のタイルに対して n+m 体以上のエージェントがタイルを掴み，n 体が同じ方向に移動すると m 体は何もしなくても n+m 体のエージェントが移動したことになる．このとき m 体のエージェントは他のエージェントに引きずられる形になる．また，エージェントにも重さを付け加えた場合，掴んでいて何もしないエージェント数を l，エージェントの重さを 1 とすると n+l 体以上のエージェントがタイルを移動させるために必要となる．



T1 は重さ1 のタイル，T2 は重さ2 のタイル

図 4.2：本研究で扱うタイルワールド

以上のように問題を設定したときの GNP が進化することでエージェントがどのように進化をしていくかの流れを図 4.3 に示す．学習は GNP の有向グラフ内にある機能を持った部分有向グラフが形成されていく，つまりサブルーチンが生成されていき，タスクを達成する機能となる有向グラフとなることで行われる．

(1) セル間を移動

まず初めにセル間を移動出来るエージェントが発生する. 開始ノードから移動する機能を持った処理ノードへのリンクが張られることを意味する.

(2) タイルを探索可能

初期位置から移動し, タイルのあるセルに移動することが出来るエージェントが発生する. タイルの場所を探索する判定ノードを移動する部分グラフにリンクを張ることでタイルを探索し, タイルに向かうことが可能となる.

(3) タイルを掴む

タイルのあるセルに入った後, タイルを掴むことが出来るエージェントが発生する. タイルを探索する判定ノードから現在位置にあるという意味のリンクをタイルを掴む機能の処理ノードへ張ることで可能となる.

(4) タイルを穴の方向へ移動

タイルを掴んだ状態でタイルを穴の方向へ移動することが出来るエージェントが発生する. タイルを掴む処理ノード実行した後に穴を探索する判定ノードを用いることでタイルを移動することが可能となる.

(5) タイルを移動させるエージェント群

複数のエージェントがタイルを掴んだ状態で穴を探索し, 同じ方向へ移動することで重さのあるタイルを運ぶことが出来るエージェント群が発生する. 4 のタイルを穴の方向に移動させることでタスクが達成されることを学習した後に複数のエージェントでのタイル運びが可能となる.

(6) 穴の位置でタイルを離すことが可能

タイルを掴んでいる状態で, 穴が存在するセルに入ったときにタイルを放すエージェントが発生する. 穴を探索する判定ノードから現在位置にあるという意味のリンクをタイルを放す機能の処理ノードへ張ることで可能となる.

(7) 他のタイルを探す

一連のタイルを落とす作業が終わった後次のタイルを探す処理を行うエージェントが発生する.

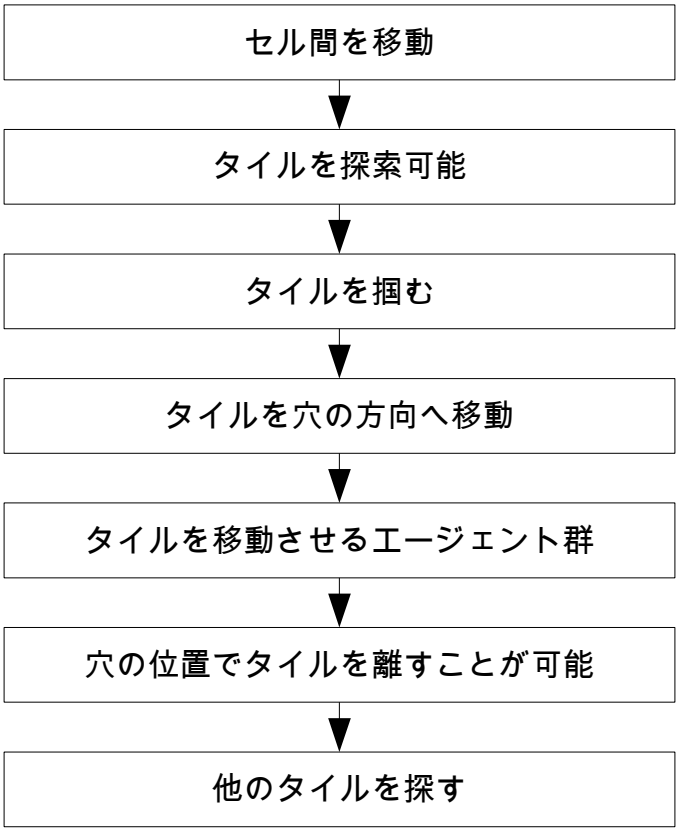


図 4.3 : 共同作業の学習過程

	1	2	3	4	5	6	7	8	9	10	11	12
1	■	■	■	■	■	■	■	■	■	■	■	■
2	■											■
3	■											■
4	■			・	・					・		■
5	■			・	・					・		■
6	■			・			・	・	・	・		■
7	■			・	・	・	・	・	・			■
8	■						・	・	・			■
9	■								・			■
10	■				A	・	・	・	・			■
11	■											■
12	■	■	■	■	■	■	■	■	■	■	■	■

・:エージェントの移動した場所

図 4.4:エージェントが移動した結果

4.3 タイルワールドにおける得点計算

タイルワールドのタイル運びによって得られる点数は，GNP による進化の評価値と適合度となる．以下は，タイルワールドでの学習の過程を元に得点の計算方法を考案した．

$$
\begin{aligned}
fitness \;=\; & C_{tile} \;\times\; \sum_{t \in TN} DropTile_t \times TileWeight_t \\
& + \; C_{dist} \;\times\; \sum_{t \in TN} \frac{(InitialDist_t - LastDist_t)}{InitialDist_t} \times TileWeight_t \\
& + \; C_{grab} \;\times\; GrabTile \\
& + \; C_{move} \;\times\; MoveArea
\end{aligned}
$$

TN はタイルの添え字の集合，$DropTile_t$ はエージェントがタイルを穴に移動させ落とした枚数，$TileWeight_t$ はタイル t の重さを表す．$InitialDist_t$ はタイルワールドの初期状態におけるタイル t と穴の最短距離，$LastDist_t$ は制限時間が過ぎた時点でのタイル t と穴の最短距離を表す．$GrabTile$ は制限時間が過ぎた時点でタイルを掴んでいるかどうかである．$MoveArea$ はタイルワールド内のエージェントの移動可能なセルに移動した数を表す(図 4.4)．C_{tile} ，C_{dist} ，C_{grab} ，C_{move} は定数で，各項目に対する重みである．

「落としたタイルの枚数」と「タイルを穴に近づける」という得点はタスクに直接関係するものであるが，「タイルを掴む」と「動いた面積」という得点はタイルを探索し掴む行動を学習させるための評価値である．この得点は直接のタスクには関係せず，また局所解に陥る原因にもなりうるが問題の構造上必要なものとして入れている．また，タイルの重さを得点に加えているのは，タイルワールド内に重さの違うタイルが配置されている時に重いタイルを優先的に運ばせるためである．本研究では簡略化のためエージェントの重さは 0 とする．

4.4 シミュレーション実験に用いたノード

シミュレーション環境に適合するように設定したノードを表 4.1 に示す. ノードの個数は, 1体のエージェントに対するものである.

表 4.1 : 実験に使用したノード

ノード名	ノードの意味	ノードの種類	処理時間	個数
GO_FORWARD	前へ1セル進む	処理	1	2
GRAB	タイルを掴む	処理	1	3
RELEASE	タイルを放す	処理	1	3
TURN_RIGHT	+90度回転	処理	1	2
TURN_LEFT	−90度回転	処理	1	2
NOP	何もしない	処理	1	2
HAVE_TILE	タイルを持っているか	判定	2	1
FIND_TILE	タイルがどこにあるか	判定	6	2
FIND_HOLE	穴がどこにあるか	判定	5	2
FIND_AGENT	他のエージェントがどこにいるか	判定	5	1
TRANSACTION_FAIL	前の行動が失敗したか	判定	2	2

ノードは 11 種類あり, GO_FORWARD, GRAB, RELEASE, TRUN_RIGHT, TURN_LEFT, NOP は処理ノードで, HAVE_TILE, FIND_TILE, FIND_HOLE, FIND_AGENT, TRANSACTION_FAIL は判定ノードである. 次に判定ノードについての分岐の意味と分岐数について以下に述べる.

HAVE_TILE は持っているか持っていないかの 2 分岐, FIND_TILE はタイルが上下左右, 今の場所にある, 存在しないの 6 分岐, FIND_HOLE は上下左右, 今の場所にあるの 5 分岐, FIND_AGENT は上下左右, 今の場所にいるの 5 分岐, TRANSACTION_FAIL は直前に行った処理ノードである GO_FORWARD と GRAB と RELEASE が失敗したかどうかの 2 分岐である.

また, ノードからノードへのリンク遅れ時間は 0 として, リンク間については制限時間を減らす操作は行わなかった.

4.5 実験環境について

実験に用いたパラメータは表 4.2 のとおりであり，遺伝的操作の選択はすべてトーナメント選択を用いた．評価値計算の重みのための定数は $C_{tile}=45$　$C_{dist}=5$　$C_{grab}=10$　$C_{move}=0.01$ とした．この場合，評価値が 100 を超えていれば1枚のタイル運びのタスクを完了したことになる．評価値が 10 を超えたときにエージェントがタイルを掴むことが出来ており，20 を超えたときにタイルを穴の存在するセルに移動出来たことを意味する．

表 4.2：実験パラメータ

世代数	500
個体数	100
制限時間	タイルの数*2000
エージェント数	2
突然変異確率	0.01
交叉確率	0.65
トーナメントサイズ	15
IGNPの記憶細胞	10

4.6 タイル1枚のタイルワールド

まず図 4.5 に示す重さ 2 のタイルが 1 枚存在するタイルワールドを用いて実験を行う．2 体のエージェントは同じ位置で，図 4.5 の A の位置に初期配置される．

	1	2	3	4	5	6	7	8	9	10	11	12
1	■	■	■	■	■	■	■	■	■	■	■	■
2	■											■
3	■		○									■
4	■											■
5	■						T2					■
6	■											■
7	■				A							■
8	■											■
9	■											■
10	■											■
11	■											■
12	■	■	■	■	■	■	■	■	■	■	■	■

図 4.5：タイル 1 枚のタイルワールド

4.6.1 共進化によるヘテロジニアスエージェントの学習

共進化によるヘテロジニアスエージェントの学習を GNP, IGNP, GNPIAM を用いて行った結果を図 4.6 に示す. グラフの横軸は世代数を表し, 縦軸は評価値を表す. 最大は 10 回のうち最終世代において最大の評価値を得た回の個体の評価値推移を示し, 平均は 10 回の各回の最大の評価値の推移を平均したものを示す.

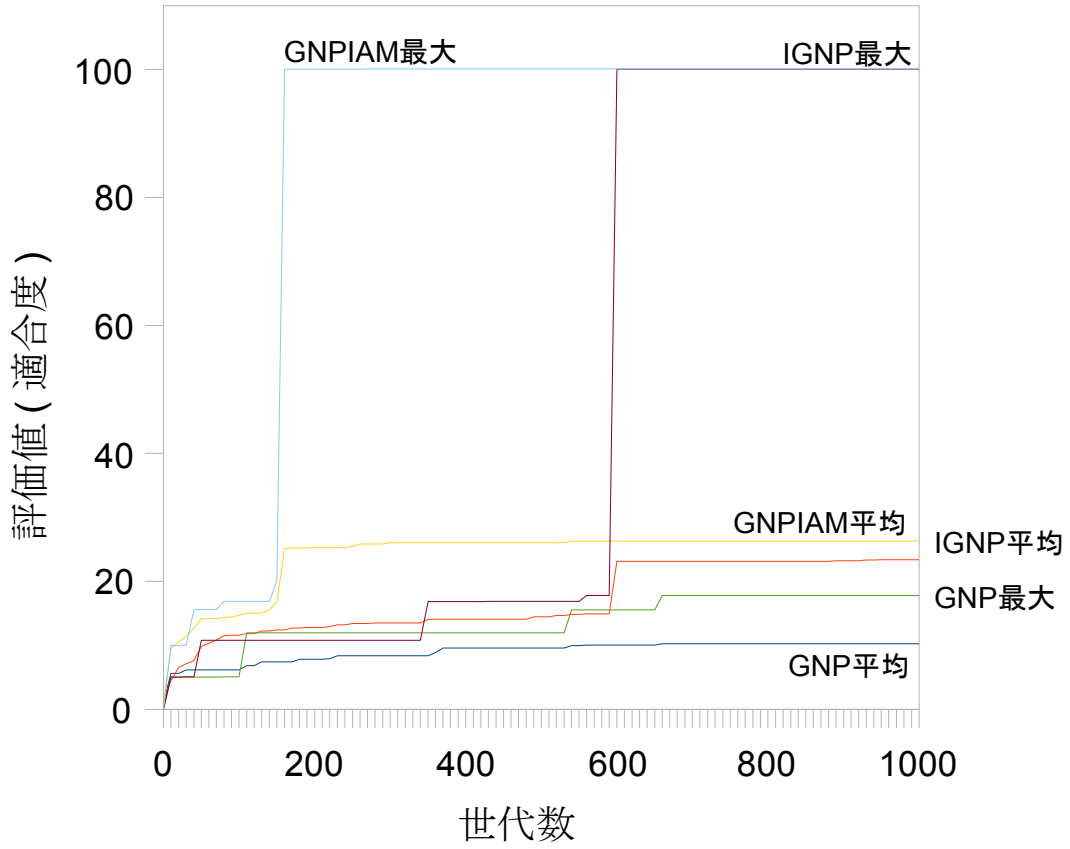


図 4.6 : タイル 1 枚のタイルワールドにおける共進化手法適用

タイル運びの学習がどの程度進んでいるかについて, 行動ごとに 10 回中何回行ったかを表 4.3 にまとめる.

表 4.3 : 学習の進度

学習行動	GNP	IGNP	GNPIAM
全エージェントがタイルを掴む	7	10	10
タイルを掴んで移動	3	8	10
穴の存在するセルまで移動	0	2	5
穴にタイルを落とす	0	1	1

4.6.2 非共進化によるヘテロジニアスエージェントの学習

非共進化によるヘテロジニアスエージェントの学習を GNP, IGNP, GNPIAM を用いて 4.6.1 と同様に行った結果を図 4.7 に示す.

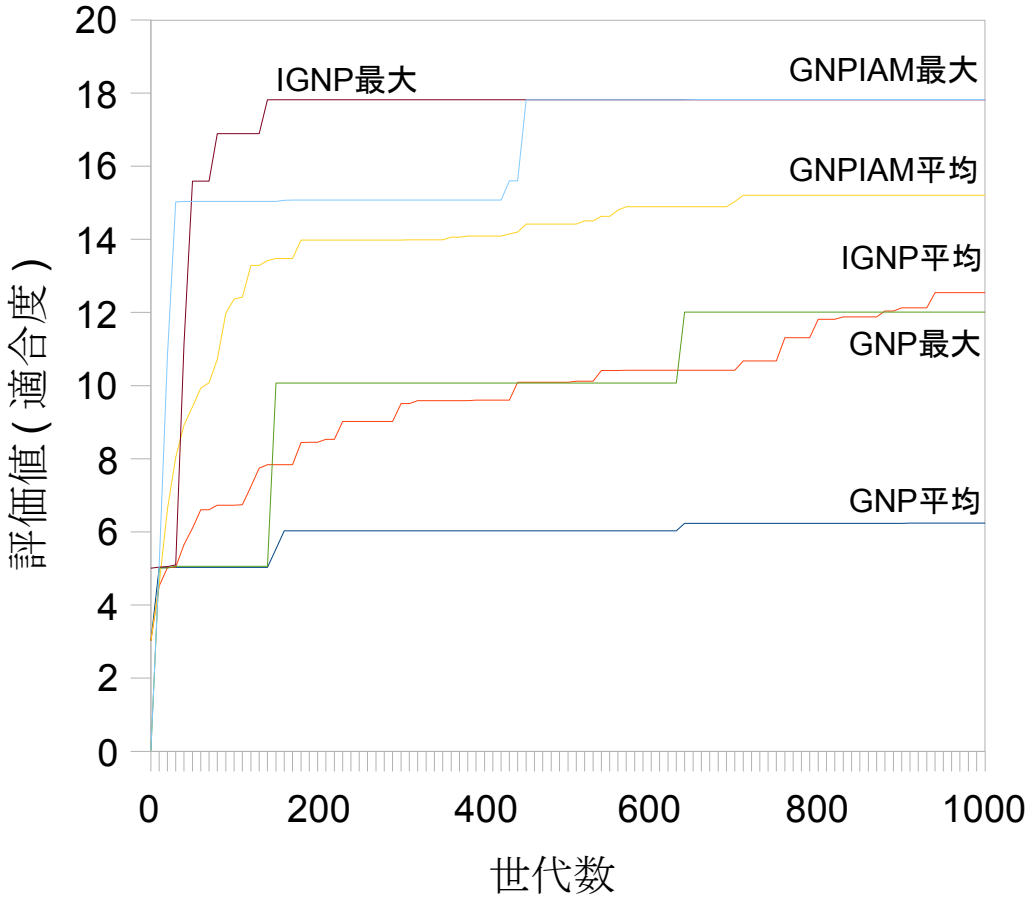


図 4.7 : タイル 1 枚のタイルワールドにおける非共進化手法適用

タイル運びの学習がどの程度進んでいるかについて, 行動ごとに 10 回中何回行ったかを表 4.4 にまとめる.

表 4.4 : 学習の進度

学習行動	GNP	IGNP	GNPIAM
全エージェントがタイルを掴む	3	9	10
タイルを掴んで移動	1	6	9
穴の存在するセルまで移動	0	0	0
穴にタイルを落とす	0	0	0

4.6.3 タイル1枚のタイルワールドの考察

1 枚のタイルを配置したタイルワールドでの各手法の実験結果の考察を行う. 図 4.6 と図 4.7 から共進化と非共進化の両方の手法に対して GNPIAM と IGNP は GNP よりも最終世代での評価値は最大, 平均共に優れていると言える. これは GNP が局所解に陥り評価値が伸びないのに対して, GNPIAM と IGNP は広い探索を行うことが出来るので良い評価値が得られたと考えられる.

GNPIAM と IGNP での評価値の違いは, 表 4.3 と表 4.4 から各学習の段階に達することが出来たかどうかの比較で考察する. GNPIAM はタイルを掴んで移動出来, 穴のあるセルまで移動することが IGNP よりも多くの回数行えるエージェントが学習できた. これは IGNP が抑制細胞や濃度によって解となる抗体をランダムに生成するのに対し, GNPIAM ではランダムではなく評価値の高い抗体をコピーした後突然変異することで生成するためである. このことはランダムで生成した場合は有向グラフ構造, つまり内部プログラムを破壊することで学習されていない抗体が生成されるあるからである.

共進化と非共進化の比較をすると, 共進化は GNP, IGNP, GNPIAM の全ての手法で非共進化の評価値を上回っていた. 非共進化の場合, あるエージェントの評価値のみでエージェント全体が行ったタスクの評価値が決定されるため評価値の突出したエージェントが生成された場合に, 初期収束と同じように評価値がそれ以上高くならなくなることが起こっている. GNPIAM と IGNP は免疫システムにより, GNP よりは改善されているが共進化よりは評価値が高くならなかった.

結果全体としては GNPIAM と IGNP で 10 回中 1 回しかタスクを達成出来ていないので, タスクを達成できる十分な学習が出来ているとは言えない.

4.7 タイル 2 枚のタイルワールド

次に図 4.8 に示す重さ 2 のタイルが 2 枚存在するタイルワールドを用いて実験を行う. 2 体のエージェントは同じ位置で, 図 4.5 の A の位置に初期配置される.

タイルワールド内に 1 つのタイルが配置された場合と, 2 つのタイルが配置された場合の違いはどのタイルを掴むかの選択の違いがある. 2 つのタイルが配置されている場合, どのタイルを選ぶかを 2 体のエージェントが協調する必要が出るので, 1 つのタイルが配置されている場合よりもタイル運びのタスクの難易度は上がる. このときに各手法においてタイル運びのタスクを行えるエージェントが学習できるかどうかの実験を行う.

	1	2	3	4	5	6	7	8	9	10	11	12
1	■	■	■	■	■	■	■	■	■	■	■	■
2	■											■
3	■						○					■
4	■		T2									■
5	■											■
6	■											■
7	■				A							■
8	■											■
9	■							T2				■
10	■											■
11	■											■
12	■	■	■	■	■	■	■	■	■	■	■	■

図 4.8 : タイル 2 枚のタイルワールド

4.7.1 共進化によるヘテロジニアスエージェントの学習

共進化によるヘテロジニアスエージェントの学習を GNP, IGNP, GNPIAM を用いて 4.6.1 と同様に行った結果を図 4.9 に示す.

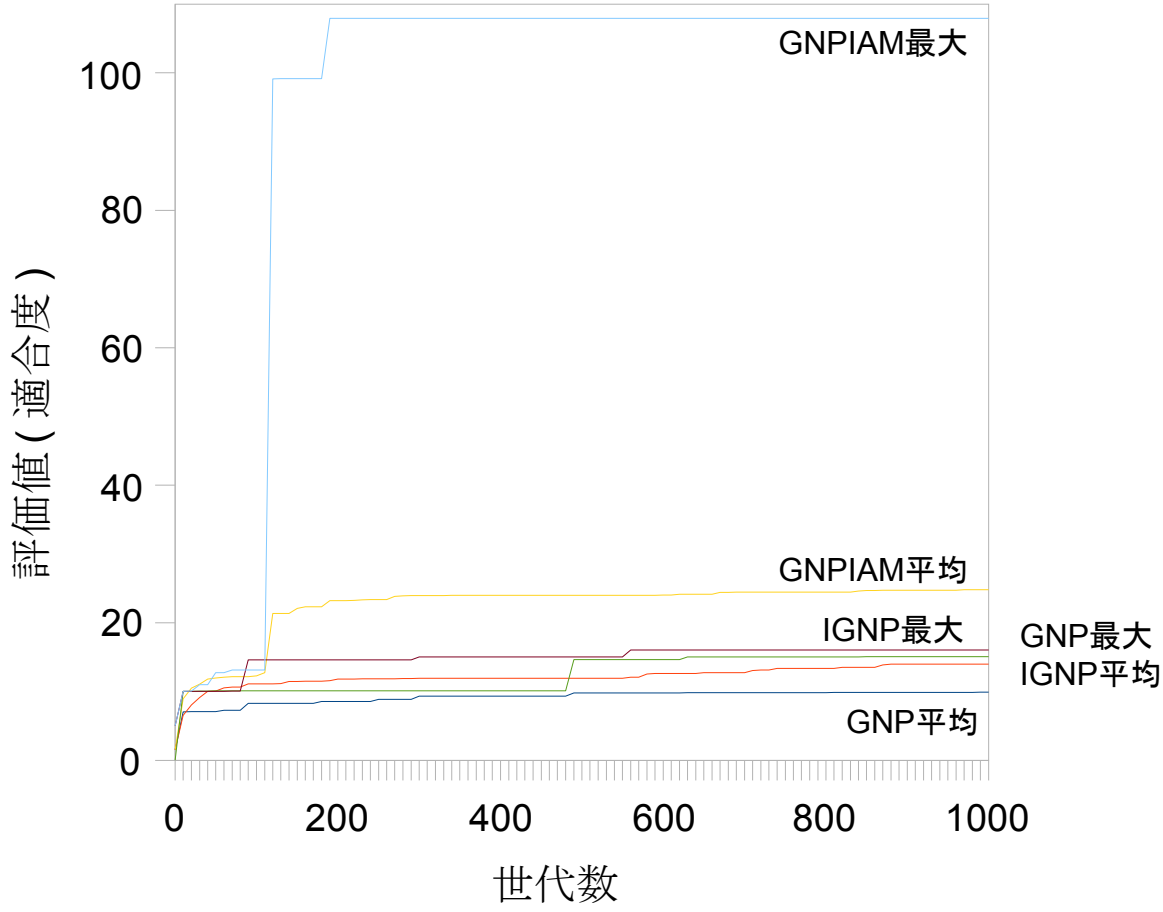


図 4.9 : タイル 2 枚のタイルワールドにおける共進化手法適用

タイル運びの学習がどの程度進んでいるかについて, 行動ごとに 10 回中何回行ったかを表 4.5 にまとめる.

表 4.5 : 学習の進度

学習行動	GNP	IGNP	GNPIAM
全エージェントがタイルを掴む	7	10	10
タイルを掴んで移動	3	9	10
穴の存在するセルまで移動	0	0	1
穴にタイルを落とす	0	0	1

4.7.2 非共進化によるヘテロジニアスエージェントの学習

非共進化によるヘテロジニアスエージェントの学習を GNP, IGNP, GNPIAM を用いて 4.6.1 と同様に行った結果を図 4.10 に示す.

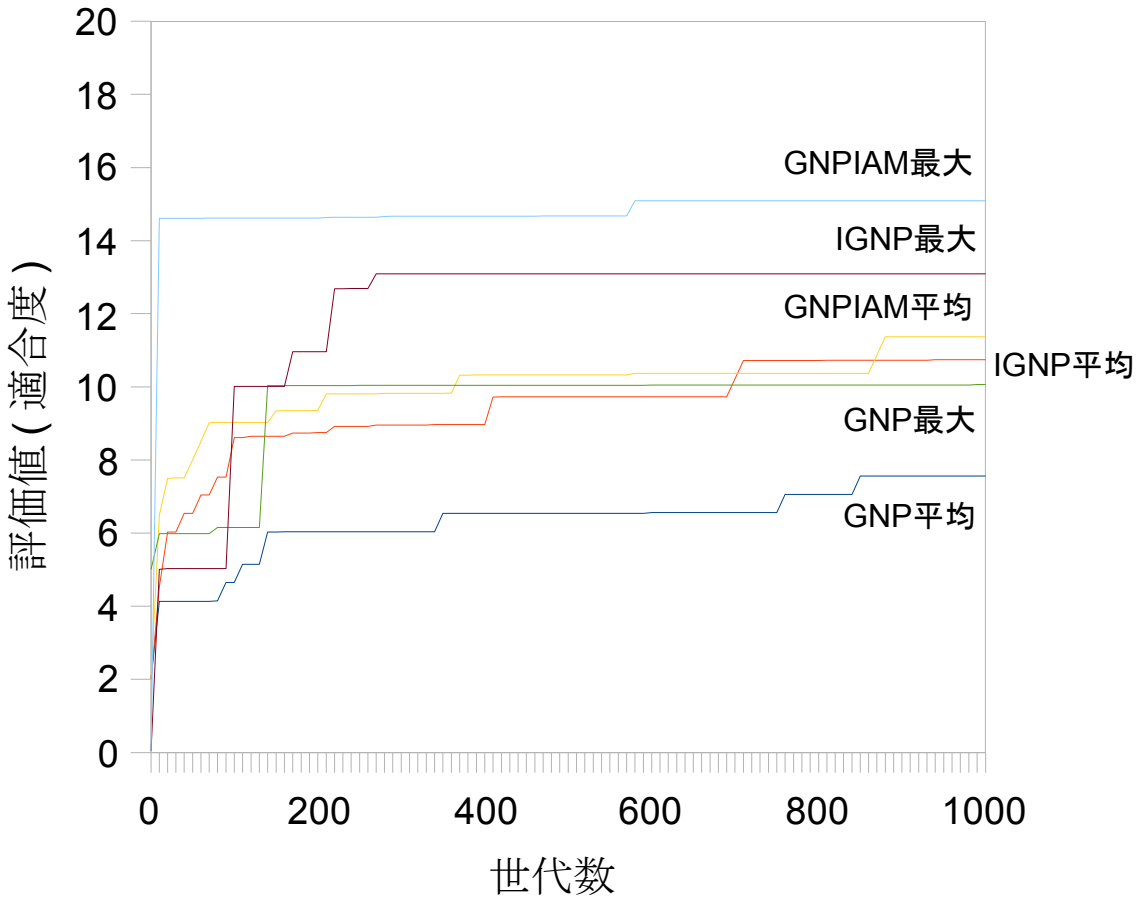


図 4.10 : タイル 2 枚のタイルワールドにおける非共進化手法適用

タイル運びの学習がどの程度進んでいるかについて, 行動ごとに 10 回中何回行ったかを表 4.6 にまとめる.

表 4.6 : 学習の進度

学習行動	GNP	IGNP	GNPIAM
全エージェントがタイルを掴む	6	10	10
タイルを掴んで移動	0	3	3
穴の存在するセルまで移動	0	0	0
穴にタイルを落とす	0	0	0

4.7.3 タイル 2 枚のタイルワールドの考察

2 枚のタイルを配置したタイルワールドでの各手法の実験結果の考察を行う. 図 4.7 と図 4.8 から GNPIAM と IGNP は GNP に比べて効率良く学習が出来ていることが分かる. GNP では 2 体のエージェントがタイルを掴むことさえも出来ていないことがあったが, 10 回中 10 回とも GNPIAM と IGNP は 2 体のエージェントがタイルを掴む, つまりタイルを探索出来ていることが分かる.

表 4.5 から共進化した場合の GNPIAM が 10 回中 1 回タイルを 1 枚落とせているが, 2 枚目のタイルは穴に落とすことが出来ていなかった. IGNP と GNP ではタイルを穴に落とすことは 10 回中 1 枚も出来ておらず, 穴の存在するセルに移動することも出来ていなかった. また, GNPIAM でも穴の存在するセルに移動させることは, 1 枚のタイルでのタイルワールドの共進化(表 4.3)と比べると少ない回数移動していることが分かる. これは 2 体のエージェントが同じタイルを掴む行動を行うことが難しいため多くの処理時間を費やし, タイルを穴の存在するセルに移動するまでに制限時間が過ぎてしまったためである. これは 2 体のエージェントが効率よく同じタイルを掴むまでの行動をうまく学習できなかったためと考えられる. 同じように非共進化の場合でもタイルを穴の存在するセルに移動出来ていない. さらに非共進化の場合はタイルを移動することさえも共進化の場合に比べて出来ておらず, 2 体のエージェントが同じタイルを掴むことを学習することも出来ていない.

4.8 シミュレーション実験についての考察

タイルを穴に落とすタスクを行うヘテロマルチエージェントの学習では, GNPIAM が評価値が一番高くなり, 次に IGNP で最低が GNP となった. この結果から GNPIAM と IGNP は GNP よりも効率的な探索を行っていることで良い学習を行うことが出来たと言える. また, ヘテロマルチエージェントを共進化した場合と非共進化した場合では共進化した場合の方が非共進化した場合よりも常に高い評価値を得ることが出来, 効率的な学習を行うことが出来たと言える.

第五章

まとめ

遺伝的に進化させることでエージェントの学習を行うことが可能な GNP を用いて，ヘテロマルチエージェントシステムを構築し共同作業を行うエージェントの学習を行い，従来手法の GNP と免疫進化手法を用いた GNPIAM と IGNP の性能比較を行った．また共同作業を行うための GNP によるヘテロマルチエージェントでは，エージェントの有向グラフ構造同士を共進化させることでマルチエージェントシステムを生成する手法と共進化させずにマルチエージェントシステムを生成する方法を提案した．

共同作業は複数体のエージェントが協調したときのみ行える作業とし，実験ではタイルワールドにおけるタイル運びの学習を行った．タイルワールドにタイルを配置し，そのタイルを穴に落とすというタイル運びのタスクに対しどの程度学習を進めることが出来たかを比較した結果，GNPIAM と IGNP はタスクを達成することが出来るエージェントを生成することが出来たが，GNP では生成することが出来なかった．このことから共同作業を行うエージェントの生成問題に対して GNPIAM と IGNP は有効に学習できることを示すことが出来た．また，GNPIAM は常に GNP よりも高い評価値を得ることが出来，GNP よりも優れた解の探索能力を発揮することが分かった．さらに，共進化させた場合のマルチエージェントシステムと共進化させない場合のマルチエージェントシステムでは，共進化させた場合が常に共進化させない場合よりも高い評価値を得ることが出来，共同作業を行うマルチエージェントシステムを生成するためには共進化させることが有効であることを示した．

今後の課題は，全ての試行で共同作業が行えるエージェントを生成することが出来ているわけではないため，現状よりも多くの割合でそのようなエージェントを生成出来るようになることである．そして，学習データとなる学習タイルワールド数を増やすことで様々なタイルワールドに対して共同作業を行えるエージェントを作成し，学習データ以外のタイルワールドにも共同作業が行えるマルチエージェントシステムを生成することが挙げられる．

謝辞

本論文を完成するにあたり，名古屋工業大学 舟橋健司 准教授には多大なるご指導をいただき心から御礼申し上げます．

研究を進めるにあたって常に御指導と御助言を与えてくださいました名古屋工業大学 伊藤宏隆 助教に深甚なる感謝の意を表します．

また，研究を進めていく上で協力をいただきました中部大学 岩堀 祐之 教授，名古屋工業大学 中村剛士 准教授，愛知教育大学 福井真二 講師に深く感謝をいたします．

参考文献

[1] 平澤宏太郎, 大久保雅文, 片桐広伸, 古月敬之, 村田純一:“蟻の行動進化における Genetic Network Programming と Genetic Programming の性能比較”電気学会論文集, 121(6), pp. 1001-1009, 2001/6

[2] 間普真吾, 平澤宏太郎, 古月敬之:”強化学習を用いた遺伝的ネットワークプログラミングの学習法とその性能評価”火の国情報シンポジウム 2004.

[3] 中越洋, 間普真吾, 平澤宏太郎, 古月敬之:“マクロノードつき遺伝的ネットワークプログラミング”電気学会論文誌 C, Vol.124C, No.8, pp.1619-1625, 2004.

[4] 間普真吾, 畠山裕之, 中越洋, 平澤宏太郎, 古月敬之:“プログラムサイズ可変型マクロノードつき遺伝的ネットワークプログラミング”電気学会論文誌C, Vol. 126, No. 4, 548-555, 2006/4

[5] 伊藤宏隆, 間瀬友裕, 岩堀祐之:“免疫型進化手法を用いた遺伝的ネットワークプログラミングによるエージェント学習”電学論C, Vol. 125, No. 4, pp.637-644 (2005)

[6] 伊藤宏隆:“大規模問題を用いた免疫調節機構を持つ遺伝的アルゴリズムの評価”第 34 回知能システムシンポジウム資料、177-182 (2007)

[7] 中村貴志, 村田忠彦:“GNP with ADG 手法によるマルチエージェントでの適切な役割分担の生成”日本知能情報ファジィ学会 ECOmp 研究部会 (名古屋, 5月28日, 2005), 発表 No.8, 6 pages

[8] 平澤宏太郎,大久保雅文,古月敬之,村田純一,松家裕子:“Genetic Network Programming によるヘテロマルチエージェントシステムの構成”電気学会論文誌C, 123(3), pp.544-551, 2003/3

[9] 江口徹, 平澤宏太郎, 古月敬之:“マルチエージェントシステムの共生進化モデルの構築”情報処理学会論文誌 - 数理モデル化と応用, 45;SIG2, pp.144-156, 2004/2