ゲームにおける戦術アルゴリズムの自動獲得*

杉山 智則, 山口 文彦, 中西 正和†
慶應義塾大学 理工学研究科 計算機科学専攻 中西研究室‡

1 要旨

計算機プログラムが取り扱う問題はより複雑に、より多様化している。それらのプログラムの中には、不確定の要素や予知し得ぬ事柄が関係して、プログラムの目標が漠然としていて明確な基準でプログラミングが出来ないものも存在する。そのような際に、その条件や状態、環境において適応したプログラムを自動生成出来れば非常に有効である。

本研究では、"XRB"(X ウインドウ上における Robot Battle) というオリジナルのモデルを用いる。XRB は 2 体の戦闘ロボットが対戦するシミュレーションゲームである。ロボットはその形態である 4 つの武装と、その武装を有効に生かす戦術プログラムによって構成される。

次に、このロボットの自動生成には、リカレント・ニューラルネットワークと遺伝的アルゴリズム (GA) の手法を用いる。リカレント・ニューラルネットワークは回帰的な結合を持つニューラルネットワークであり、時系列パターンを扱うことができる。また、GA は自然淘汰と突然変異によっての進化をプログラムに応用しようと考えられた多点探索アルゴリズムである。

このリカレント・ニューラルネットワークと GA を複合して用い、より環境に適応したロボットを自動生成するのである。XRB ではより環境に適合するとは、他のロボットを倒し、負けなくなり、高勝利率を得ることである。

有効な戦術プログラム及び、ロボットの形態を早期に効果的に獲得する方法を検討し、その有効性について評価する。

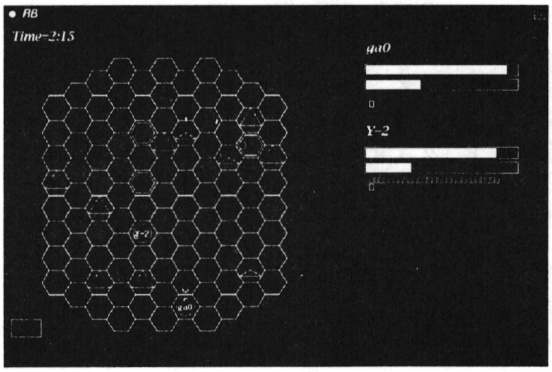

図 1: XRB

2 環境モデル XRB

研究を行なう環境として、山口が製作した XRB を用いる。XRB は 2 体の戦闘ロボットが対戦するシミュレーションゲームである。

2.1 XRB の概要

XRB では、ロボットが行動を決定するためのプログラムをそれぞれ 1 つ持ち、それに従って判断と行動を繰り返していく。勝敗は、ゲームスタートからある時間内に一定のダメージを越えると、そのロボットは破壊される。また、規定の時間経過した際にダメージの総量が一定値を越えていないときはより攻撃的であったことを評価し、移動距離の長いロボットが判定勝ちとなる。ロボットのプログラムは、ロボットを制御するための拡張命令セットを加えた T11

*Program Synthesis for Robot Battle Game using Genetic Algorithm and Recurrent Network

†Tomonori SUGIYAMA, Fumihiko YAMAGUCHI, Masakazu NAKANISHI

‡Nakanishi Laboratory, Department of Computer Science, Faculty of Science and Technology, Keio University.

アセンブラプログラム、もしくは、ロボット制御ライブラリを加えたC言語で記述される。そのプログラムを自動生成し、より勝率の高いロボットプログラムを獲得することが、本研究の目的である。

2.2 Rule

ロボットは、図1に示すヘックスによって構成されたフィールド内で、自由に動き回ることができる。

ロボットの持つ武器は表2のようになっており、各ロボットは重複を許し4種類の武器を装備できる。武器の中にはロボットの移動速度や、探査範囲などのロボットの能力が優れるようなものも存在している。

2.3 命令サブセット

ロボットを制御する命令を表3に記す。ロボットの動作を制御する命令とロボットのセンサを制御する命令がある。

3 Genetic Algorithm

3.1 GAの概要

自然淘汰と突然変異によっての進化をプログラムに応用しようと生まれてきたのが遺伝アルゴリズム (Genetic Algorithm GA) である。

GA は問題を遺伝子と適合度関数で表現し、複数探索点に対して、選択、交叉、突然変異を繰り返し、世代が進むことにより収束していくという多点探索アルゴリズムである。

3.2 システム構成

前章で述べたとおりT11アセンブラでロボットの動作は記述されている。したがって、本研究では、評価されるべき個体はT-11アセンブラプログラムであるといえる。そのT-11アセンブラプログラムをオブジェクトファイルになおし、XRBで他のロボットたちとの対戦を実行する。その結果をGAシステムは得る。

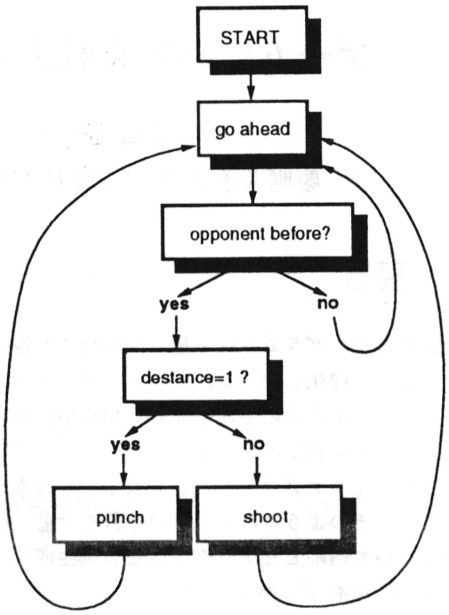

図 5: Tree Stracture

3.3 コード化

本システムでは評価されるべき個体はT-11アセンブラプログラムであるといえる。そこで、そのプログラムに対応した遺伝子コードを用意する必要がある。

ここでXRBにおけるオペレーションについて考えてみると表3におけるACTIONとJUDGEの2種類に分類できる。ACTIONは分岐のない行動であり、JUDGEは分岐する各種条件による判断である。

これらのオペレーションの組合せによってプログラムは書かれており、その組合せは木構造で表すことができる。

3.4 GAオペレーション

3.4.1 Fitness

次世代の個体群に対して大きく影響する評価値は他のロボットとの対戦結果を用いた。

weapon	bullet	damage	range	comment
stabilizer	無制限	−	−	旋回が速くなる
engine	無制限	−	−	移動が速くなる
radar	無制限	−	−	探査範囲が広くなる
armor	50 ダメージ	−	−	装甲によって防御力が増す
turlet	無制限	−	−	左右斜め前方に射撃が可能となる
thruster	無制限	−	−	斜め移動が可能となる
charge	無制限	5 ×移動 Hex	前方移動	突撃をかけることが可能となる
cannon	40	10	前方	通常砲弾
vulcan	100	2	前方	威力は弱いが連射性のよい弾
shotgun	20 × 5	6 × 5	前 90 度	5 発同時発射する広がる弾
atomic	1	200	前方	強力だが 1 発のみ
boomerang	20	20	左右前方	60 度づつ旋回する弾
laser	30	10	前方	弾速は速いが威力が弱い弾
mine	20	50	真下	敵にのみ反応する地雷
missile	10	40	前方	近接信管を持ったミサイル
spray	8	32	左右前方 3Hex	広角度、近距離弾
knuckle	無制限	40	前方 1Hex	拳で殴る
whip	無制限	25	前方 2Hex	鞭で攻撃

図 2: 武器一覧

ACTION	
forward	前進する
back	後退する
left turn	左旋回する
right turn	右旋回する
shoot	弾を撃つ
JUDGE	
search enemy	敵を探査する
search bullet	敵弾を探査する
search land	地形を探査する
check enegy	残りエネルギーを調べる
check damage	ダメージを調べる
check time	残り時間を調べる
check rest bullet	残弾数を調べる

図 3: 行動一覧

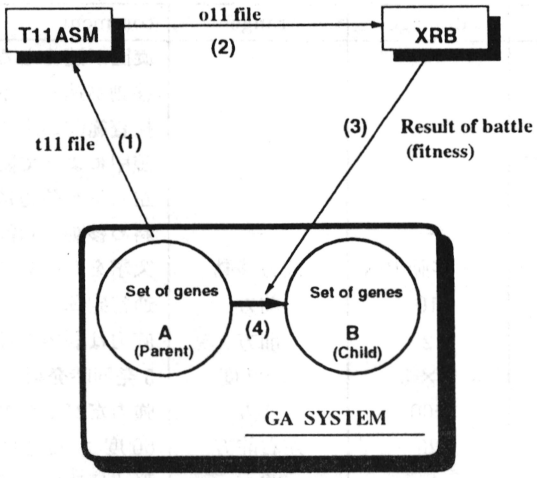

図 4: GA System

3.4.2 Cross-over

複数の親遺伝子を組み合わせて次世代の遺伝子を作る操作であるクロスオーバは木構造の任意の 2 箇所を選び、枝から下を付け変えることによって行なう。

3.4.3 Mutation

突然変移の操作は以下の 3 種類が考えつく。

- ジャンプ先を変更する
- オペレーションの内容を変更する
- 新しい木構造を生成し、付け変える

4 リカレント・ニューラルネットワーク

4.1 リカレント・ニューラルネットワークの概要

リカレント・ニューラルネットワークは入力層に出力層からのフィードバック信号が送られるという特徴を持つネットワークである。時刻 t における入力は、入力層の x_t と時刻 $t-1$ における出力層の y_{t-1} をフィードバックさせた信号である。この信号は、時刻 t における出力に関与するのは時刻 $t-1$ における出力と時刻 t における外部入力である。

$$Y_k(t) = \sum_i w_{ki} X_i(t) + \sum_j u_{kj} Y_j(t-1)$$

$Y_i(t)$ ：時刻 t における出力層 i ユニット
$X_i(t)$ ：時刻 t における入力層 i ユニット
w_{ij} ：入力層 j から出力層 i に送られる際のウエイト
u_{ij} ：出力層 j から出力層 i に送られる際のウエイト

時刻 $t-1$ の出力を入力として用いるので、このリカレント・ニューラルネットワークは一次の記憶を持つ必要がある。

4.2 システム概要

リカレント・ニューラルネットワークを用いたシステムは、各センサーによって得られる情報を入力として入力層に入れ、出力層のユニットそれぞれにオペレーションを対応させ、最も大きな出力をこのネットワークの t におけるオペレーションとして実行する。仮に入力が一定の値しか入ってこないと仮定すると、リカレント・ニューラルネットワークは試行によらず、必ず同様の出力列を出力し、分岐のないプログラムと見なせる。また、入力が変化すると入力

という条件によって分岐するプログラムと見なせる。

4.3 ジャンプ命令

リカレント・ニューラルネットワークのシステムをプログラムと見なしたとき、ジャンプ命令をより明確に実現するために以下のオペレーションを追加する。

catch 現在の出力ユニットの値を全て記憶する

throw 全てのユニットを対応する catch によって記憶した値にする

reset すべてのユニットを初期化する

全ての出力ユニットを記憶することにより、ある時刻 t におけるプログラムカウンタを記憶し、throw によって、過去のある時刻 t に戻ることにより、ジャンプを表現する。また、全てのユニットを初期化することによって時刻 0 に戻ること、つまりプログラムの先頭に戻ることがが可能である。

5 実験

実験は以下のテーマに基づいて行なった。

5.1 実験設定

実験 1 GA において違う環境においての対応

これは異なる環境においても、目的のプログラムを自動生成し、より勝率の高いロボットプログラムを獲得することが可能であるかということである。本研究の場合の環境とは対戦相手のプログラム群であり、目的のプログラムは強いプログラムのことである。

実験 2 GA において次世代の生成方法の比較

次世代は、現在の世代からクロスオーバー、ミューテーションという操作を経て作られる。それぞれの効果を知るために両方をおこなったもの、クロスオーバーのみでミューテーションを行なわないもの、逆にクロスオーバーを行わず、ミューテーションのみのものという 3 種類について実験を行なった。

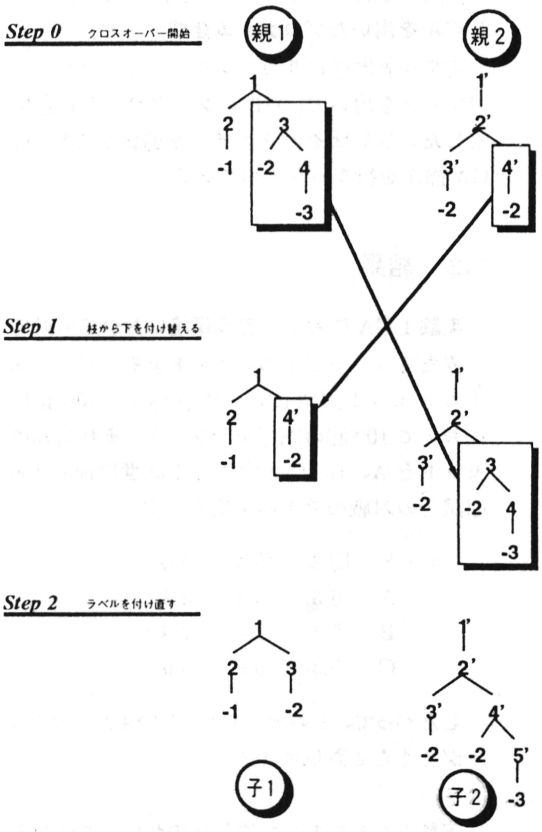

図 6: Crossover

実験 3 GA において不確定な環境要素における対応

本研究の場合の環境とは対戦相手のプログラム群であるが、それ以外に不確定の要素が入って来た場合の反応を調べる。今回の実験ではその要素として、小さめな要素としてランダムで障害物を発生を用い、大きめな要素としてスタート位置をランダムにした。

実験 4 リカレント・ニューラルネットワークを用いたプログラム生成

リカレント・ニューラルネットワークを用いて目的のプログラムを自動生成し、より勝率の高いロボットプログラムを獲得することが可能であるかを実験したものである。

実験5 リカレント・ニューラルネットワークとGAを用いたプログラム生成

実験4と同様にリカレント・ニューラルネットワークを用いて、目的のプログラムを自動生成した。GAはそのウエイトを遺伝子と扱い、GA操作を行なったものである。

5.2 結果

実験1 GAにおいて違う環境においての対応

異なるコンセプトのロボットをそれぞれ5体ずつ3セット用意して、40個体で、100世代において100回の試行を行なった。それぞれのセットをA、B、Cとすると100世代後の100回試行の対戦の勝率の平均を記す。

セット	勝ち	負け	分け
A	6.3	0.3	3.4
B	5.1	2.0	2.9
C	5.3	0.8	3.9

したがって、どのセットも十分効果的なロボットができたと評価できる。

実験2 GAにおいて次世代の生成方法の比較

実験1と同様に、クロスオーバー、ミューテーションの両方、クロスオーバーのみ、ミューテーションのみの3セットにおいて40個体で100世代、100試行を行なった。

セット	勝ち	負け	分け
両方	5.3	4.1	0.6
Cross-overのみ	6.8	2.0	1.2
Mutationのみ	2.1	6.3	1.6

この結果をみると、Cross-overは非常に有効であるがMutationは逆にマイナス効果になっているようである。それは効果的なMutationが行なわれる確率が非常に小さく改悪をしているのである。

実験3 GAにおいて不確定な環境要素における対応

この実験では不確定な要素として障害物とランダムスタートポジションの行なった。

セット	勝ち	負け	分け
通常	5.5	1.3	3.2
障害物あり	1.9	4.1	4.0
Random Set-up	1.1	7.0	1.9
両方あり	1.4	5.2	3.4

不確定の要素が絡んでくると良い結果が現れなかった。障害物が入ると引きわけが増えるのは2体が遭遇しない確率が増えるからである。より不確定要素に対応できるようにするには世代をもっと多く繰り返す必要がある。

実験4 リカレント・ニューラルネットワークを用いたプログラム生成

リカレント・ニューラルネットワークのウエイトを乱数で生成し、実験1と同様に、異なるコンセプトのロボットをそれぞれ5体ずつ3セット用意して、40個体で、100世代において100回の試行を行なった。それぞれのセットをA、B、Cとし、100回試行の対戦の勝率の平均を結果とした。ただし、淘汰のみを行ない他のGAの操作は行なわず、淘汰された遺伝子の代わりは初期個体と同様に乱数で新しいウエイトを生成した。

セット	勝ち	負け	分け
A	0.5	8.9	0.6
B	0.3	9.3	0.4
C	1.1	8.0	0.9

特定の一部分では効果的な動作をすることもあったが、全体としてはまったく有効であるとはいえなかった。

実験5 リカレント・ニューラルネットワークとGAを用いたプログラム生成

実験4とほぼ同様であるが、クロスオーバー、ミューテーションの操作を行なった。遺伝子として扱っているものは、各ユニットは結ぶリンクのウエイトの全てである。

セット	勝ち	負け	分け
A	0.4	9.3	0.3
B	0.5	9.2	0.3
C	0.7	8.7	0.6

実験4と同様にまったく有効ではなかった。クロスオーバーもウエイトを換えるという行為であるためほとんど親の特性を継承できなかった。

6 結論

実験の結果、ロボットの戦術の獲得にGA手法は効果があることが分かった。また、リカレント・ニューラルネットワークを用いた手法には、その有効性について論じるだけの実験を現在行なえていない。その用い方、学習のさせ方等の検討が必要である。

7 展望・発展の方向性

個々のプレイヤーが作成したプログラムに打ち勝つ戦術を自動生成することは、ゲームの奥深さに影響し、発展の方向についてもそれぞれ違う方向に向かう可能性がありその魅力ははかりしれない。しかし、現在は有効にはたらいていないのが現実であり、より効果的に生成するための方向性を示唆する。

7.1 リカレント・ニューラルネットワークのコンパイル

現在用いているリカレント・ニューラルネットワークは全てのユニット状態をプログラムカウンタとし、また、時系列に進むそれらのユニットの状態遷移がプログラムであるといえる。そのユニットの変化をその度に計算するいまの形はインタプリタであるといえる。各ネットワークにおいて初期状態と各ユニットをつなぐウエイトは決定しており、入力層は有限であるので全ての状態を事前に計算しておくことは可能である。それをなんらかの形で記憶し、実際の入力によって実行ができれば、もとのリカレント・ニューラルネットワークと同等のプログラムにコンパイルできたといえるであろう。コンパイルすることのメリットはいうまでもなく高速化なのであるが、それ以外にも記憶領域の縮小化等の可能性があると思われる。

7.2 リカレント・ニューラルネットワークのサブルーチン化

現状はプログラム全体をリカレント・ニューラルネットワークを用いて作成している。しかし、求めたいプログラムが非常に大きくそれができる可能性が低いと思われる。そこで、それぞれのリカレント・ニューラルネットワークをサブルーチンとして扱い、メタな存在としてそれらの複数のサブルーチンへ分岐する共通のフレームを用意する。そして、複数のリカレント・ニューラルネットワークのサブルーチンの複合によって求めたいプログラムを表現する。この方法は複数のリカレント・ニューラルネットワークによって一つのプログラムを表すのであるから、記憶領域に関してはその複数倍になってしまうが先に述べたコンパイルを行なうことにより可能であると思われる。

7.3 リカレント・ニューラルネットワークのGAによる学習

現システムもGAで学習を行なっているのであるが、各ユニット間のウエイトを変化させることは非常に効果が大きく、また効果的な操作を行なえない。つまり、GA操作がほとんど有効に働いていないと思われる。しかし、サブルーチン化することによって、それぞれのサブルーチンで分割できるので、そのそれぞれのまとまりを一つの遺伝子として扱えばGAによる学習が効果的に働くと思われる。

参考文献

[1] Goldberg, D. E. *Genenetic algorithms in search, optimization, and machine learning* Addison-Wesley, 1989.

[2] Holland, J. H. *Adaptation in natural and artificial systems* The University of Michigan Press, 1975.

[3] John R. Koza, *Genetic programming* The MIT Press, 1992.

[4] Sugiyama Tomonori, *Program Synthesis for Robot War using Genetic Algorithm* AAAI-94 Workshop on Artificial Intelligence, Artificial Life, and Entertainment, pages 22-26, 1994.

[5] Grefensteffe, J. J. et al. *Genetic algorithms for the traveling salesman problem* Proc. of an International Conference on Genetic Algorithms and Their Applications, pp.160-168.

[6] R. Axelrod and D. Dion. The Further Evolution of Coopration, Science, vol. 242, pages 1385-1390, December 1988.

[7] K. Lindgren. In *Evolutinary Phenomena in Simple Dynamics* In C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, Artificial Life II. SFI Studies in the Science of Complexity, 1991.

[8] S. W. Wilson: Knowledge growth in an artificial animal, In K.s. Narendra, editors, *Adaptive and learning systems: Theory and applications.* Plenum Press, 1986.

[9] 荻原拓也. 遺伝的アルゴリズムを用いたリカレントニューラルネットワークの学習. 情報処理学会, 第 49 回全国大会, September 1994.