The 32nd Annual Conference of the Japanese Society for Artificial Intelligence, 2018

遺伝的アルゴリズムによるシューティングゲームにおけるゲーム AI

An AI game player trained by a genetic algorithm that avoids bombardments in a shooting game

滑川　静海[*1]　　手塚　太郎[*2]
Shizuma Namekawa　　Taro Tezuka

[*1] 筑波大学知識情報・図書館学類
College of Knowledge and Library Sciences, University of Tsukuba

[*2] 筑波大学図書館情報メディア系
Faculty of Library, Information and Media Science, University of Tsukuba

Making programs play games has contributed significantly to the advancement of AI research. It is partly because popular games often resemble real-world problems. Tackling them has enabled AI to cope with real-world scenarios as well. Now that AI has surpassed humans in strategy board games such as chess and go, one next target would be to train it to play video games. This paper focuses on a shooting game and optimizes a program to avoid bombardments deployed by the opposing player. Using a genetic algorithm, the AI player was optimized to move around without hitting enemy attacks. The results of experiments showed that it can successfully learn to do so, although with much computation time.

1. はじめに

人工知能の発展と共に、それにゲームをプレイさせる試みが盛んである。人間がゲームを好むのは本能的な欲求と親和性があるからであり、ポピュラーなゲームは人間の生存に必要なタスクと何らかの類似性を持っていることが多い。そのためゲームで人工知能を鍛えることは今後、人工知能を現実社会に進出させていく上で有益であると考えられる。ゲームは現実世界のタスクと比較してシミュレーションが容易であり、その学習プロセスを研究しやすいというメリットもある。

現在、ゲームに使われている AI には様々な種類が存在している。大別するとゲーム自体をコントロールするメタ AI、ゲームの NPC（non-player character）として機能するキャラクターAI、その時々に応じたゲームの進行を支えるナビゲーション AI の三種類に分けることができる。今回はそのうちのキャラクターAI に分類されるものを扱う。

本研究では AI 技術のうち、特に遺伝的アルゴリズムを用いて戦略を学習させるアプローチをとる。遺伝的アルゴリズムとは生物が進化していく仕組みから着想を得た最適化アルゴリズムである。多数の個体のそれぞれが解の候補を持ち、それらが突然変異や交叉と呼ばれる形で値を変化させながら、最適解を探索する。また、損失関数が小さい個体ほど自らの複製を多数生成するため、その解候補周辺で重点的に探索が行われる。

使用したゲームは PC を中心に生産されている東方 project シリーズの東方妖々夢（図 1）というゲームである。これはいわゆる弾幕シューティングゲームというジャンルに該当する。弾幕シューティングとは、敵機が大量の弾を発射し、自機がそれを避けつつ敵を倒していくというゲームを指す。

熟練した人間のプレーヤーはシューティングゲームにおいて敵が放った大量の弾幕を驚くほどの器用さで回避できる。本研究の目標は遺伝的アルゴリズムを用いてそのようなプレーを AI にも実行させることである。

連絡先：滑川静海，筑波大学知識情報・図書館学類，〒305-855
茨城県つくば市春日 1-2，s1511536@u.tsukuba.ac.jp

このような能力の実現は、たとえば混雑した交通状況の中で自動運転の車をスムーズに移動させる機能にも繋がることが期待される。

図1：東方妖々夢のスナップショット

2. 関連研究

2.1 ゲーム AI

ゲーム AI は様々なジャンルのゲームで研究開発されている。将棋や囲碁といったパズルゲームだけでなく RPG やアクションゲームにも応用されてきている。例えば、昔のアクションゲームは、敵の出現位置や強さが固定されていた。しかし現在はオンライン学習という技術が研究されている。Marc Ponsen（2004）によると「オンライン学習とは、ゲームがリリースされた後に、AI がプレイ中に最適な行動を選択していくというものである。」としている。これはどういうことかというと、以前までのゲームでは完全にランダムにするか決められた環境で遊ぶことしかできなかった。しかしこのオンライン学習という技術を用いると、ゲームの進行中に AI が自分のプレイに適応する、つまり自分の実力にあった難易度や自分のリズムにあったプレイスタイルでゲームを進行することができるようになる。

他の AI を用いた研究としては、AlphaGo が挙げられる。これは碁のプログラムであるが、ニューラルネットワークやディープラーニング、強化学習などを用いて学習をしている。特徴的な学

習方法として、囲碁の AI 同士で対戦することによってお互いを高め合う形で強化学習をしている。これは遺伝的アルゴリズムにおいても応用可能性があり、研究として扱うことができるだろう。

2.2 遺伝的アルゴリズム

遺伝的アルゴリズムに関しても様々な研究がなされている。遺伝的アルゴリズム自体が物事の最適化を進めることのできるものであるため、ゲーム分野との親和性が高い技術と言える。したがってゲーム分野において有効に応用することができる。例えば、本研究で扱うようなシューティングゲームや、アクションゲーム、RPG など多岐にわたる分野で使うことができる。三宅(2008)によると「ゲーム内のキャラクタの進化などに使用される」としている。具体的には、いくつも個体を用意し、それらに何度もトラップを課すことで進化させるというものである。他にもレースゲームに遺伝的アルゴリズムを適用することで効果を得た例もある。このように遺伝的アルゴリズムはゲーム AI の実現に高い適合性を持っていることがわかる。

3. 提案手法

3.1 遺伝的アルゴリズムの概略

本研究で提案する手法では、プレイヤーキャラクターを操作して条件をクリアする個体が現れるまで繰り返し解の探索を続けることである。具体的には、上下左右と動きなしの五種類のコマンドを遺伝子として持った個体を 30 体用意し、各個体でステージをプレイさせ、その個体の中から親候補となる 10 個体をエリート選択方式で選択する。そして選ばれた10個体の中から2個体をルーレット選択方式で選び交叉を行い、3体の個体を生成する。これを10回繰り返すことで新たな世代の30個体を生成する。以上の一連の操作をクリア個体が出るまで繰り返し行い、その経過を見る。

3.2 遺伝子データ

今回は左右のコマンドが選ばれる確率を 5%、上下が選ばれる確率を3%とし、動かない確率を残りの84%とする。これらにより作られた遺伝子データを持つ個体を 30 体用意する。今回の実験ではゴールまでにかかるコマンド数が未知数であるため、また交叉の質を保つため第一世代や上の世代のコマンド数を超えた場合は逐次コマンドを上記の確率に基づいて生成する。この際、なぜ上記のような偏りのある確率にしたかというと、シューティングゲームにおいて、時期が動くことはリスクが高いからである。通常プレイするときは、飛んでくる弾を避ける際にむやみに動かず、自分に当たると予測される球を見つけた際に動いて避けるという動作を行う。従って、動き続けるのではなく動くべき時に動く AI が実用的かつ人間味のある動きと言える。このような理由からそれぞれのコマンドに偏りを持たせている。また、左右と上下の確率に違いがあることもシューティングゲームの特徴が起因していて、東方 project は縦方向シューティングであるため敵の弾は上から飛んでくることになる。その弾を避ける際、縦方向に動くことよりも横方向に動いて避ける方が避けやすいということは感覚的にも明らかだろう。以上のような理由からある程度試行錯誤した結果今回の確率分布に設定した。

3.3 ゲームのプレイ

用意した 30 体の個体を実際にプレイさせる。ここでのプレイ方法であるが、実際に売られているゲームであるためソースコードが公開されていない。そのような理由で UWSC という OS マクロツールを用いて、実際にゲーム上でキー操作を行う形でコマンド操作を行う。

この実験におけるゲーム内の設定であるが、自機の残機は 1(一度被弾した時点で死亡)とする。また、弾は常に発射している状態でボム(一定時間無敵かつ強力な攻撃)は使わないという制約のもと操作を行う。また、実験を行う際のゲームの難易度は easy として進める。

今回の実験においては、第一世代に限り、コマンド生成→コマンド入力→死亡判定→コマンド生成…のように繰り返す。第二世代以降は、引き継いだコマンドを入力→死亡判定の操作を繰り返すが、引き継いだコマンド数を超えて生存していた場合、第一世代と同様にコマンド生成フェーズを挟む。以上のようにゲームのプレイを行い、ひとつの個体がゲームオーバーになる度にそのログを保存する。

3.4 遺伝子の評価と選択、交叉

ゲームのプレイにより得たログを元に遺伝子の評価を行う。今回は単純にステージで最も長く進んだ個体、つまり終了時のコマンド数が大きい個体ほど優秀とする。これによりそれぞれの個体に優秀さの序列をつけることができる。この序列を用いて親の遺伝子を決定するが、今回の実験ではエリート選択方式とルーレット選択方式を掛け合わせた形で選択を行う。具体的には、先に親候補となる遺伝子上位 10 体を選択する。その後、選ばれた 10 個体の中からルーレット選択により親となる二個体を選択する。これにより選択された親二個体から二点交叉で二個体、一様交叉で一個体の計三個体を生成する。

ここで二点交叉は複数交叉に分類されるが、複数交叉とは、北野(1991)によると、「交叉位置が複数ある方法である。例えば、交叉位置が 4 と8なら、新たな個体 B の 5 番目から 8 番目まで、個体 A の 9 番目から最後までによって、遺伝子が作られる。同時に、その逆の組み合わせで、もう一つの新たな個体の遺伝子が作られる。」としている。これは図2、図3を用いて説明する。図2が交叉前の遺伝子を持つ個体 A、B であり、交叉位置としてランダムに選択された二点に囲まれた部分を入れ替えることで、新たな二つの遺伝子を持つ個体(図3)を生成する。また、一様交叉とは、北野(1991)によると「交叉時にマスクをかけてそれによってどちらの親の遺伝子を受け継ぐかを決定する方法である。」としている。これは図4で示している。今回はどちらの遺伝子を引き継ぐ確率はどちらも等しく 1/2 としている。

以上の一連の操作を子孫が 30 体になるまで、つまり 10 回繰り返す。これにより次の世代が決定する。

ここで一つ問題となる点がある。それは、遺伝子の長さが一定ではないため交叉をする際に存在しない点で交叉をしてしまうことがある点である。これを解消するために、交叉をするフェーズに入ったタイミングで遺伝子の大きい個体のコマンド数に合わせて、小さい個体にコマンドの確率を条件通りに保った状態で新たなコマンドを追加する。以下で表を用いて説明する。長さ 9 の遺伝子を持つ個体 1(表 1)と、長さ 3 を持つ個体 2(表 2)を親として選んだ場合を考える。

表における矢印はコマンドの入力された向き、S は stay の S でその場から動かないコマンドを指す。また、×は被弾したことによりゲームオーバーになった状態を示している。

この二つの個体で交叉を行う際に、個体 2 においては 4 以降のコマンドが抜け落ちてしまっている。この状態で交叉を行ってしまうと、表 3 のように被弾した後にコマンドが入力されうるという、ありえない遺伝子データが出来上がってしまう可能性がある。これを解消するために、交叉直前の個体 2 の遺伝子に個体1と比

べた不足コマンド数を条件で指定した確率で追加する。つまり、表 4 のように遺伝子の大きさが等しくなるようにコマンドを補充した後に交叉を行う。

これは最初に遺伝子の長さを一定にした状態で交叉することと本質的には変わらない操作であるため、実験上問題にはならない。

最後に、生成された 30 個体に突然変異を施す。突然変異が起きる確率は 1%と設定し、突然変異に選ばれた子孫個体は、ランダムに選ばれたコマンド二つが入れ替わる。

以上の操作により次世代を構成する子孫を生成する。

表 1:個体 1 のコマンド系列

Index	1	2	3	4	5	6	7	8	9
Command	←	S	S	S	↑	→	S	S	↓

表 2:個体 2 のコマンド系列

Index	1	2	3	4	5	6	7	8	9
Command	↑	S	S	×	×	×	×	×	×

表 3:個体 1 と個体 2 を交叉してできた遺伝子の一例

Index	1	2	3	4	5	6	7	8	9
Command	↑	S	S	×	↑	×	S	S	×

表 4:個体 2 にコマンドを補ってできた遺伝子の一例

Index	1	2	3	4	5	6	7	8	9
Command	↑	S	S	S	←	S	S	S	↑

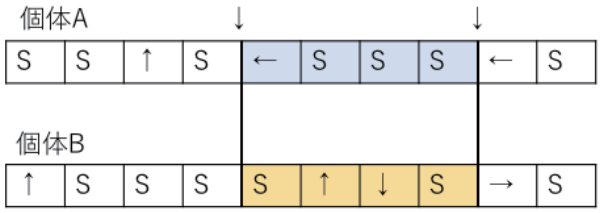

図 2:二点交叉前の個体 A、B

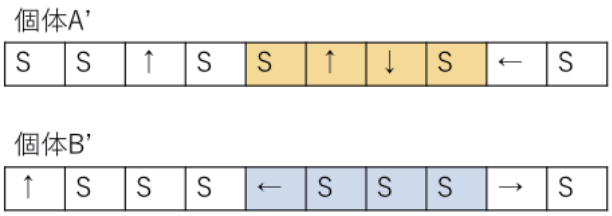

図 3:二点交叉後の個体 A、B

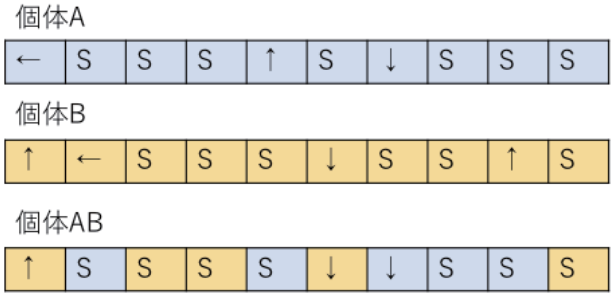

図 4:一様交叉

4. 実験結果

今回は自機が目標とする地点まで到達できるようなコマンド系列が得られることを終了条件とし、終了までに 76 世代を要した。実験により得られた各世代のコマンド数の平均値を図 5 に示した。コマンド数とは遺伝子の長さを指す。

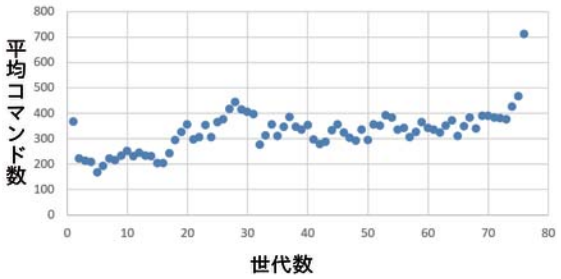

図5:遺伝的アルゴリズムの進行に伴う各世代における平均コマンド数の変化

平均コマンド数を見ることによって学習が進んでいるかを見ることができる。今回の実験においては、平均コマンド数が伸びているならば、よりステージの先に進んでいる個体が多いことを示す。図5から分かるように全体としては徐々に学習していき、右肩上がりになっている。今回の実験で特徴的だった世代としては、第 1 世代、第 28 世代、第 76 世代である。これらの世代では平均コマンド数の局所的なピークが現れている

まず第 1 世代に焦点を当てると、平均値が第 2 世代に比べて高いが、原因としては外れ値として高いコマンド数を持つ遺伝子がいたため、平均値が高くなっていたことが挙げられる。しかし第 2 世代になった途端に一気に平均コマンド数が落ちてしまった。この原因に関してはシューティングゲームの特性が関わっている。シューティングゲームというのは、プレイ中連続的に自機が動いていく、言い換えると、前の瞬間の状況を引き継いだ上で次の行動をとる。つまりこの場合、コマンド一つ一つが独立しているわけではないので、一つ前のコマンドが変わってしまうだけでそこより後のコマンド全てに影響が出てしまう。この特性のために第2世代では平均コマンド数が落ちてしまったと考えられる

第28世代も同様のことが言え、偶然28世代付近では上手くいっていたが、まだ収束するに至っていなかったために次以降の世代では落ち込んでしまったと考えられる。

最後に規定の場所に達した第76世代であるが、これは平均が急激に高くなっている。この理由としては、第75世代における最優秀個体が多く選ばれ、かつ運良く追加されたコマンドが上手く噛み合う形になったからということが考えられる。第69世代から第73世代にかけて横ばいで、かつ第74世代から伸びていることから、この辺りで第73世代までの部分が収束していた可能性が考えられる。遺伝子が収束することで、収束した部分までは同じ動きをするため、その部分までは安定的に到達することができる。従って平均コマンド数の伸びることにつながる。以上のような理由で第76世代において急激に平均コマンド数が伸びたと考えられる。

5. 結論

本研究ではシューティングゲームにおいて敵が打つ弾幕を回避する戦略を遺伝的アルゴリズムを用いて学習させた。

実験の結果、シューティングゲームにおける遺伝的アルゴリズムの適用は有効であった。しかし同時に様々な問題点や課題点が浮上してきた。

一つ目に挙げられるのは、その学習にかかる膨大な時間である。一般にプログラムを動かすのと異なり、60FPS という条件下でコマンドの入力を行うため、ソースコードの有無にかかわらずプレイさせるのに時間を要する。従って、シューティングゲーム、あるいは、より複雑なゲームの AI に適用することを考えると、相応の量の学習時間を見積もる必要がある。それを防ぐためにGUI 上で操作しなくても済むシミュレートツールを作ることが解決策として挙げられる。

次に挙げられる問題点としてはシューティングゲームという特性上遺伝子を組み替えることのリスクが大きいということである。これは 3 でも説明した通り、コマンド一つ一つが独立していなく連続であるがゆえのリスクである。これのために交叉をした際に、一つ前の世代と比較して平均コマンド数が大きく落ちてしまうという現象が見られた。今回は親の選択にエリート選択とルーレット選択を掛け合わせたものを用いたが、やはりこれだけでは良い個体を失ってしまうリスクがあるため、エリート選択で最低でも 1 個体は最優秀のものをそのまま残すという形をとった方が良いと考えられる。

以上のような問題点を解決することができれば学習効率が大幅に上昇し、よりゲーム AI の発展につながるだろう。また、RPG やパズルゲームなどではニューラルネットやディープラーニング等の技術と遺伝的アルゴリズムを組み合わせた AI を組み合わせた AI を作る研究も行われており、シューティングゲームにおいてこれらを利用することも今後の課題である。

謝辞

本研究は, JSPS 科研費 JP16K00228, JP16H02904 の助成を受けたものである。

参考文献

[Marc Ponsen 2004] Marc Ponsen: Improving Adaptive Game AI with Evolutionary Learning , Faculty of Media & Knowledge Engineering Delft University of Technology , 2004.

[伊瀬 顕史 2011] 伊瀬 顕史, 馬野 元秀, 瀬田 和久: Tuning of Fuzzy Rules with a Real-Coded Genetic Algorithm in Car Racing Game, The 27th Fuzzy System Symposium, 2011.

[三宅 洋一郎 2008] 三宅 陽一郎 : How to Use AI Technologies to Develop Digital Games, 人工知能学会誌, 23 巻1号, 2008 年 1 月.

[北野 宏明 1991] 北野 宏明: 遺伝的アルゴリズム, 人工知能学会誌, 7 巻 1 号 , 1991.

[Michael Genesereth 2005] Michael Genesereth, Nathaniel Love, Barney Pell: General Game Playing: AI Magazine, 62-72, 2005.

[David Silver 2016] David Silver, Aja Huang, Chris J. Maddison: Mastering the game of Go with deep neural networks and tree search, Nature 529, pp.484-489, 2016

[Pieter Spronck 2006] Pieter Spronck, Marc PonsenIda Sprinkhuizen-Kuyper: Adaptive game AI with dynamic scripting, Machine Learning, Vol. 63, No. 3, pp. 217-248, 2006.

[川野 洋 2006] 川野 洋: シューティングゲームの敵機攻撃弾発射アルゴリズムに関する考察, 情報処理学会研究報告, 2006.

[片岡 俊 2007] 片岡 俊: 遺伝アルゴリズムを用いたシューティングゲームの攻撃弾発射手法, 高知工科大学 情報システム工学科, 2007.

[Georgios N. Yannakakis 2018] Georgios N. Yannakakis and Julian Togelius : Artificial Intelligence and Games, Springer, 2018.

ZUN: 東方妖々夢, http://www16.big.or.jp/~zun/html/th07.html(2018/03/08)

umiumi, “UWSC”, UWSC, http://www16.big.or.jp/~zun/html/th07.html(2018/03/08)