Vol.2016-GI-35 No.13
2016/3/9

StarCraft AI における Deep Q-Network を用いたユニットコントロールの実装

中田 季利　新井 イスマイル

概要：StarCraft: BroodWar は、Blizzard Entertainment が開発したリアルタイム・ストラテジーゲームである。大きな状態空間と複雑なゲームルールから、AI Competition が開催されるなど研究対象として注目されているゲームである。既存の戦闘 AI における弱点として、不利な地形への誘い込みへの弱さや、味方ユニット間の位置関係の悪さが指摘されている。既存の強化学習を用いた手法では、地形や味方ユニットとの位置関係を最大限利用できていないためと考えた。しかし、地形や味方ユニットの位置関係等を考慮すると状態空間が大きくなってしまい学習が難しい。これを解決するため、Convolutional Neural Network(CNN) を活用し大きな状態空間上で直接 Q 学習が可能となる手法である Deep Q-Network(DQN) の適用を試みた。CNN にマップの侵入可否情報を入力して、上下左右の移動方向もしくは攻撃を選択する DQN を作成して学習を行った結果、マップを理解して移動ができ、報酬に基づき敵ユニットから逃げる行動が学習されることを確認した。

キーワード：ゲーム, ゲームにおける学習, マルチエージェントゲーム, 機械学習

1. はじめに

StarCraft: Brood War[*1] は、Blizzard Entertainment が開発したリアルタイム・ストラテジーゲーム (以下 RTS ゲーム) である。RTS ゲームは広大な状態空間上の非完全情報ゲームであり、またリアルタイムで迅速な意思決定が必要である。このため、人工知能の研究対象として大きく注目されている。2010 年から StarCraft AI Competition[*2] という大会も開かれ、2014 年には 18 チーム登録されるなど大きな盛り上がりを見せている。AI 開発を容易にするために BWAPI[*3] といった API も公式に公開されている。

StarCraft は、ターン制ではなくリアルタイムに進行し、戦場の指揮官となって兵士 (ユニット) を指揮して戦うゲームである。予め幾らか用意されている戦場 (マップ) 上で対戦を行う。プレイヤーは Terran・Zarg・Protoss の 3 つの種族から 1 つを選択し、その種族からなるユニットを操作し戦うことになる。ユニットには数種類あり、その種類ごとに初期の HP・エネルギー・アーマー・攻撃力などが割り当てられている。HP が 0 になるとユニットが破壊され、勝利とは全ての敵ユニットの HP を 0 にすることである。

StarCraft は、探索・生産・建設・戦闘などさまざまな要素が絡んでいる複雑なゲームになっているが、本研究では戦闘 AI の改良に焦点をあてている。既存の戦闘 AI における弱点として、不利な地形への誘い込みへの弱さや、味方ユニット間の位置関係の悪さが指摘されている。既存の手法では、地形や味方ユニットとの位置関係を最大限利用できていないためであると考えられる。しかし、ルールベースでそれらを調整するのは難しく、また機械学習を適用するにも状態空間が広大という問題があった。一方、Convolutional Neural Network(CNN) を活用し大きな状態空間上で直接 Q 学習が可能となる手法である Deep Q-Network(DQN) という手法が他のゲームの AI に用いられるようになってきている。これを踏まえ、StarCraft の戦闘におけるユニットの行動である、上下左右の移動方向もしくは攻撃の選択に DQN を適用した実装を行った。その場に留まっていると不利な戦闘になってしまうシチュエーションで 125 万イテレーション学習を行った結果、地形を理解し、角を曲がるように敵ユニットから逃げるような行動が学習された。

本論文では、2 章にて、研究対象となる Starcraft の概要について述べ、3 章で Starcraft のユニットコントロールにおける関連研究と既存手法を提示し、現状の課題について検討する。そして 4 章で本研究の提案手法を述べ、5 章ではその実験結果と考察を行い、6 章にてまとめる。

[*1] http://us.blizzard.com/en-us/games/sc/
[*2] http://www.sscaitournament.com/
[*3] http://bwapi.github.io/

2. StarCraft について

StarCraft は、ターン制ではなくリアルタイムに進行し、戦場の指揮官となって兵士 (ユニット) を指揮して戦うゲームである。予め幾らか用意されている戦場 (マップ) 上で対戦を行う。敵ユニットを全て倒し勝利することが目的になる。対人戦では、長時間のゲームで数百ものユニットを人間が操作する、大規模な対戦も繰り広げられている。

プレイヤーは Terran・Zarg・Protoss の 3 つの種族から 1 つを選択し、その種族からなるユニットを操作し戦うことになる。各種族には以下の特徴がある。

- Terran
  地球人の末裔。他の種族と違い無条件で建造物を建造することができる。SCV というワーカーユニットによって、建造物や機械系ユニットを資源を消費して修理することができる。
- Zarg
  超文明種族によって創造された有機体種族。ユニットの生産コストが低いため、数と機動力で攻めていくスタイルが一般的である。全ての建造物が Drone という Zarg のユニットが変異したものであり、ダメージを自然回復することができる。Terran と違い建造物は Creep と呼ばれる生物組織が覆っている地表の上にしか建てることができず、Creep を広げながら領地を広げていくことになる。生産所から全ての種類のユニットを生産することができ、柔軟な戦術をとることができる。
- Protoss
  Zarg と同じく超文明種族によって創造された種族。各ユニットの能力が高く、少数精鋭で戦う。ユニットと建造物は HP とは別に自然回復するシールドを持っており、HP へのダメージを肩代わりできる。Pylon という建造物のエネルギーが届く範囲にしか建物を建設できない。

射程内に敵ユニットが存在する時、そのユニットに攻撃して HP を減らすことができる。ただし、武器にはクールダウン時間が存在し、一度攻撃するとクールダウン中は攻撃できない。HP が 0 になったとき、そのユニットは消滅する。HP やクールダウン時間、移動速度などはユニットの種類によって異なる。

ユニットには固有の特殊能力であるスキルが割り当てられている。例えば Terran のユニットである Medic のもつスキル Heal は、範囲内にある傷ついたユニットを回復することができる。また、同じく Terran の Vessel のもつスキル Defensive Matrix は、ユニット 1 体に HP250 分のダメージを肩代わりするシールドを与えることができる。

多種多様なスキルによってユニットは性格付けられており、さまざまな種類のユニットやスキルを上手く使うことが、戦闘を有利に進める上で重要である。

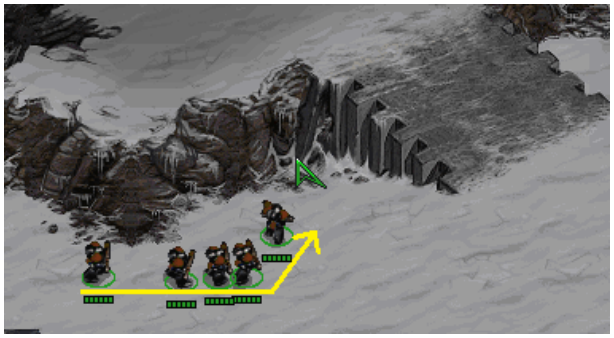


図 1 複数ユニットが直線的に移動している様子

StarCraft は、戦闘の他に以下に示す探索・生産・建設などさまざまな要素が絡んでいる。

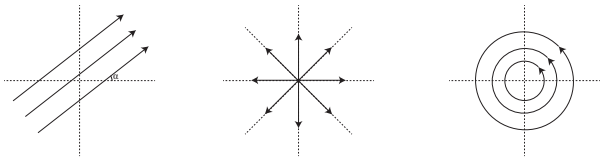
- 探索
  ゲーム中プレイヤーは、味方ユニット付近の情報しか得ることができない。偵察機などで偵察し、敵の状況や資源の探索をする必要がある。
- 生産
  マップ上に存在する有限資源である、ミネラルやガスを採掘して、味方ユニットを生産したり、研究を進めてユニットを強化することができる。
- 建設
  資源を消費して、マップ上に建造物を建設することができる。建造物には、ユニットを生産するもの、人口制限を緩和させるもの、ユニットの防御のために使うもの、スキルを研究して有効化するものなど様々な種類がある。

3. 関連研究

3.1 陣形を考慮した StarCraft AI

StarCraft の戦闘におけるユニット制御に、武田家の八陣形のひとつである横陣隊列を取り入れる手法 [1] が報告されている。実際のゲームではほとんどの場合で複数ユニット対複数ユニットの対戦となる。敵ユニットの数だけ火力が分散してしまうため、同じ性能のユニット同士での戦闘では、戦闘に参加しているユニット数がとても重要な要素になる。ユニットは重なりあうことができないため、敵を細い道に誘い込んで、広い出口から迎え撃つような戦術も存在する。これらからわかるように、戦場として選ぶ場所取りやユニットの位置関係は勝敗に大きく起因する要素である。

また既存の StarCraft AI の分析として、**図 1** のように直線的な移動をしがち、誘導行動に弱いなどの弱点が指摘されている。敵の居る方向に兵士を垂直方向に広げて配置する横陣という陣形を取り入れた AI は、陣形を考慮せず直線的に移動する AI より強いことが実験で示されているが、狭い通路では横陣が乱れてしまい勝率が下がることも



一様流れ　湧き出し　渦糸
$w(z) = Ue^{-i\alpha}z$　$w(z) = Q\log z (Q > 0)$　$w(z) = -i\Gamma \log z$

図 2　主なポテンシャル流れ

報告されている。

3.2　ポテンシャル流れを利用した AI

ポテンシャル流れを利用して、敵ユニットを囲むように味方ユニットを配置する手法 [2] が提案されている。一様流れ・湧き出し・渦糸の 3 種類のポテンシャル流れを重ねあわせることによって複雑なポテンシャル場をつくり、その微分によってユニットを移動させる。

敵軍ユニットの中心に向かって一様流れを設置することによって、自軍ユニットは敵軍ユニットに近づいていく。また、各敵軍ユニットに湧き出しをつけることによって、敵に近づきすぎない斥力をつくる。これによって、自軍が敵軍を囲んでいるような陣形になる。これは、敵軍が一部のユニットしか攻撃できないのに対して、自軍は全ユニットが攻撃できるため有利である。

しかし、多くのユニットが参加する大規模な戦闘では、ユニットとユニットの衝突が発生しやすい。他のユニットに邪魔されて目的地に到達できないユニットが発生することがある。そこで、他のユニットをブロックするユニットに渦糸を設置することによって、うまくユニットを避けて移動することができる。

この手法によって、様々な AI を相手にした際に高い勝率を得られることが示されている。しかし、実験には障害物が存在しない広いマップが使用されている。建造物などを含む複雑な地形のような、より現実的なシチュエーションに対応する必要があると結論付けられている。

3.3　強化学習を用いたユニット制御

強化学習によって、Kiting と呼ばれる追いかけてくる敵と一定距離を保つ戦法を学習する手法 [3] が提案されている。

Kiting が成立するためには、敵ユニットより素早く移動できることと、敵ユニットより攻撃射程が長いことが必要である。この能力優位を使い、敵ユニットに攻撃されない場所へ移動しながら攻撃してダメージを与えていく。

one-step Q-learning や Watkins's $Q(\lambda)$、one-step Sarsa、Sarsa(λ) を用いて学習し、比較している。

学習に用いる状態として、以下のパラメータを用いる。

- 自分の武器のクールダウン値
- 最も近い敵ユニットへの距離
- 攻撃が届く敵ユニットの数
- 自分の HP

また、以下の 2 つの行動を選択することを学習する。

- 攻撃行動
  攻撃が届く敵ユニットの中で、最も HP が低いものに攻撃を行う。
- 退却行動
  危険回避を優先して、敵のダメージを受けないように回避する。

学習中に与えられる報酬 r は、以下の式で定義される。

$$r_{t+1} = \sum_{i=1}^{m} enemy\_health_{i_t} - enemy\_health_{i_{t+1}} - (agent\_health_t - agent\_healt_{t+1}) \quad (1)$$

Terran の地上ユニットである Vulture が、Marine 6 体に囲まれているような初期状態からの Vulture の操作を学習する。

1000 エピソードの学習の結果、どの学習手法でもビルトイン AI に対して 100%に近い高い勝率を得ることができた。この結果は、RTS ゲームの AI に強化学習を用いることへの希望的な結果だとしている一方、学習をより早くすることや、Kiting に限らない一般のゲーム状況に対応するためには更なる研究が必要であるとされている。

3.4　UCB を用いたモンテカルロ木探索

モンテカルロ木探索を改良した UCT(UCB applied to Tree) という手法でユニットコントロールを決定する手法が提案されている [4]。モンテカルロ木探索とは、可能な着手を始点として敵味方共に完全にランダムな行動をした際のシミュレーションを大量に行い、その結果の期待値を評価値とする探索手法である。UCT では UCB(Upper Confidence Bound) という評価値を計算し、これが最も大きい着手を優先してシミュレーションする。i 番目の着手の UCB は、以下のように計算する。

$$UCB(i) = Q_i + C\sqrt{\frac{\ln N}{N_i}} \quad (2)$$

ここで、Q_i は i 番目の着手の評価値、C は定数、N はノード i の親における試行数、N_i はノード i の試行数である。第一項はその着手が良い成果を残せる可能性が高いほど高い値となり、第二項は他のノードと比較して探索数が少ないほど高い値となる。探索があまりされていないが期待できる着手を優先して選択するため、良いノードを効率良く探索できる。ここで C を大きく設定すると未だに探索が充分にされていない、評価値の不確定性が高いノードの UCB が高まる。そのためそれらのノードが優先的に探索されるようになり局所解に陥りにくくなるが、良い解である可能性の低いノードについて探索される回数が多くなっ

てしまう。

i 番目の着手の j 回目のシミュレーションにおける評価値 $q_i^{(j)}$ は、以下のように計算する。

$$q_i^{(j)} = \omega_1 HP + \omega_2 DM + \omega_3 CP + \omega_4 EG \qquad (3)$$

HP、DM、CP、EG はそれぞれユニットの残体力、敵に与えるダメージ、移動速度、残エネルギーを示している。また、ω_n はそれぞれ手動で設定する係数である。Q_i はシミュレーション毎に q_i の期待値となるように更新する。

実際に可能な行動は、移動・スキルの使用であり、これのどちらかが選択される。リアルタイムで行動し続けなければならないゲームのため、探索を適当な時間で打ち切り、その時点での評価値が最も高いものが次の着手として選択される。この手法において、評価値にはユニットの持つパラメータしか考慮されていない。実際の戦闘では、ユニットが作る戦線や近辺にある建造物の位置など、考慮すべきパラメータはより多くあるため、この評価値の計算は不十分だと考えられる。

3.5 Deep Q-Network

Deep Q-Network を用いて Atari 2600 のゲームプレイを学習した結果 [5] が報告されている。Deep Q-Network は、Q 学習における行動価値観数 $Q(s,a)$ を CNN によって近似することで、非常に大きな状態空間に Q 学習を適用する手法である。また、Experience Replay という、行動の結果を即時に学習に反映せずに Replay Memory に溜めておき、そこからランダムに取り出したサンプルを用いて学習する手法も取り入れている。CNN とは、2 次元グリッドの情報を 2 次元のまま畳み込みを繰り返して、グリッド上の相対位置によらない特徴の抽出を行うニューラルネットワークである。Atari 2600 のゲームにおいて、最新 4 フレームのゲーム画面を 110x84 の解像度にダウンサンプリングし、さらにグレースケール化した画像を入力に与える。出力はコントローラーへの入力の行動価値となる。各イテレーションでは最も評価値が高い行動を選択するが、一定確率で行動価値を無視して完全にランダムな行動を選択する epsilon-greedy という手法が使われている。また、報酬は各ゲーム毎に定義された獲得スコアを与えている。

Deep Q-Network の詳細なアルゴリズムは以下となる。

(1) Replay-Memory D をある大きさ N で初期化する
(2) ニューラルネットワークをランダムな重みで初期化する
(3) 各イテレーションについて
  (a) 観測された初期状態 $s_1 = \{x_1\}$ からエンコードされた入力 $\phi_1 = \phi(s_1)$ を作る
  (b) $t = 1$ から T までについて (ただし s_T がゲームの終了状態とする)
    (i) ϵ の確率で a からランダムな行動 a_t を選択する
    (ii) それ以外の場合は、$Q^*(\phi(s_t), a; \theta)$ が最大となるような a_t を選択する
    (iii) 行動 a_t を実行し、報酬 r_t と次の入力 x_{t+1} を得る
    (iv) s_t, a_t, x_{t+1} を用いて入力 $\phi(t+1)$ をエンコードする
    (v) $(\phi_t, a_t, r_t, \phi_{t+1})$ を D に格納する
    (vi) D からランダムに 1 つの minibatch である 4 つ組 $(\phi_j, a_j, r_j, \phi_{j+1})$ を取り出す
    (vii) minibatch と下式で表される y_j を用いて Q 値を更新する

$$y_j = \begin{cases} r_j(\text{終了状態の場合}) \\ b(\text{終了状態でない場合}) \end{cases}$$

このように構築された Deep Q-Network で 1000 万イテレーション分学習を行った結果、Breakout、Pong、Enduro では人間に勝利できている。これらは比較的単純なゲームの成果であるが、 Space Invaders のように比較的長期的な戦略や先読みが必要なゲームでは良い成績にならないことが報告されている。Deep Q-Network を用いて、ロボットの移動制御などを行う応用 [*4] などが発表されており、さらなる発展が望まれるが、StarCraft や他の RTS ゲームに適用された報告はまだなされていない。

以上の関連研究から、戦闘において地形や味方ユニットとの位置関係が重要であり、それらを考慮して行動を決定することが必要だと考え、その改善のために Deep Q-Network を利用することを考えた。

4. 提案手法

4.1 概要

戦闘において、不利な地形にとびこないことや、有利な立ち位置で攻撃することは重要である。しかし機械学習を用いる既存のアルゴリズムでは、HP や武器のクールダウンタイムなどの入力は用いているが、マップの情報は状態空間が大きいため有効に使うことができていない。そこで、DQN を用いてそれらの情報をグリッドマップとして与えることによって、マップの情報を活かした行動の学習ができるのではないかと考えた。

それを踏まえて、以下の 2 手法を提案する。

4.2 学習方法 1

図 3 のように、地形情報のみで CNN を構成する。ユニット情報は座標と情報をセットにした組として、CNN とは別でニューラルネットワークに与えることにする。以

[*4] http://research.preferred.jp/2015/06/distributed-deep-reinforcement-learning/

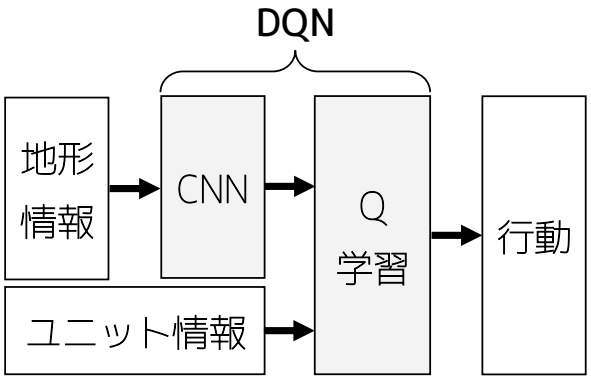


図 3 システム概要

下の方法で、DQN への入力を生成する。

(1) あるユニットを中心に 32x32 セルの大きさのグリッドを戦闘空間として扱う。1 セルの大きさは 8x8 ドット (ユニットの最小移動単位) である。

(2) 各セルに、地形の侵入可否を表すグリッドを作成する。これを CNN に入力する。

(3) 戦闘空間上のユニット毎に座標・HP・クールダウン値の情報を、Q 学習への入力として加える。

また、以下の 9 行動のうち 1 つを選択することを、DQN の出力とする。

- 縦横斜めの各 8 方向へ移動
- その場に留まって、一番近い敵に対して攻撃する。

時間 t における、ユニット i に与えられる報酬 $reward(i,t)$ は、以下の式で定義される。ただし、$cause\_damage(i,t)$ はユニット i が時間 t に敵に与えたダメージを正規化したもの、$unit\_health(i,t)$ はユニット i の時間 t における HP を正規化したものである。

$$\begin{aligned} unit\_reward(i,t) =& cause\_damage(i,t) \\ & - \{unit\_health(i,t) \\ & - unit\_health(i,t+1)\} \qquad (4) \\ reward(i,t) =& \frac{2}{3} unit\_reward(i,t) \\ & + \sum_{j \neq i} \frac{1}{3} unit\_reward(j,t) \qquad (5) \end{aligned}$$

4.3 学習方法 2

epsilon-greedy のみで複雑なゲームのプレイを学習することは難しい。初期の学習を促進させるために、ルールベースの AI を補助的に用いることを考えた。

方法 1 と同じように DQN への入力を生成し、報酬を与える。ただし、以下の 2 行動のうち 1 つを選択することを、DQN の出力とする。

- **攻撃的行動**
  敵の方向へ移動し、攻撃できるなら最も近い敵に攻撃をする
- **防御的行動**
  敵から逃げるように移動する

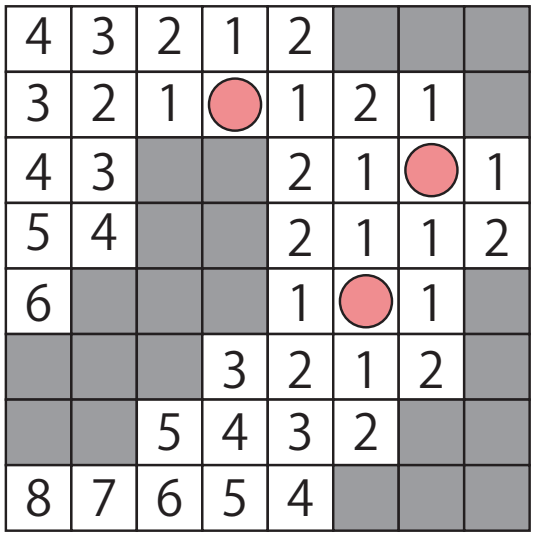


図 4 $D_{enemy}(x,y)$ の例

4.3.1 移動方向の決定

図 4 のような、グリッドマップ上のセルに対して敵ユニットへの最短距離を表す関数 $D_{enemy}(x,y)$ を考える。セル内の数値は $D_{enemy}(x,y)$ の値を表しており、灰色は壁、赤色の丸は敵ユニットの位置を表している。

攻撃的行動では、最も近い敵に近づけるような方向 ($D_{enemy}(x,y)$ が最も小さくなる方向) に移動する。このとき、自分の位置から A*探索によって移動方向を決定できる。もし、射程内に敵ユニットが存在して攻撃ができる状態なら、移動を行わずに攻撃を行う。

防御的行動では、攻撃的行動とは逆に敵ユニットへの最短距離が大きくなる方向 ($D_{enemy}(x,y)$ が最も大きくなる方向) に移動をする。

4.4 実験環境

図 5 のようなマップを用意する。青色が敵ユニットである Marine 8 体で、赤色が味方ユニットである Marine 8 体である。味方ユニットの初期位置は狭い通路になっており、その位置で敵を迎え撃つと不利になってしまう、というシチュエーションを想定している。

5. 実験結果及び考察

現段階では、提案手法のうち学習方法 1 までの実装と実験が完了しているので、本論文では学習方法 1 についてのみ結果と考察を報告する。

本研究では、Intel Corei7 6700K、Palit NE5XTIX015KB-PG600F (GTX TITAN X 12GB) の計算資源を用い、Windows 10 上で動作するプログラムを作成した。Starcraft の制御プログラムは C++と BWAPI を用いて作成し、学習プ

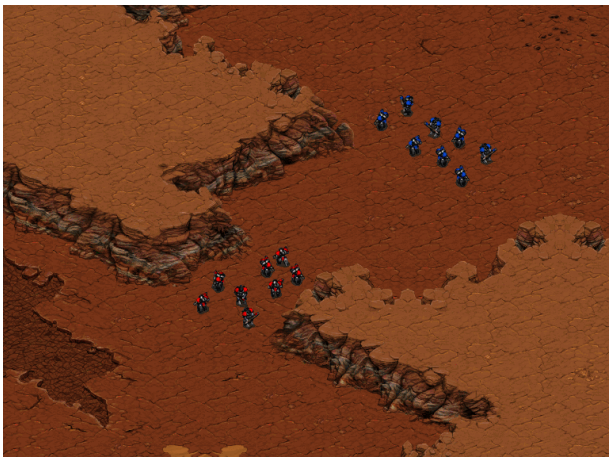


図 5 ユニットの初期配置

図 6 逃げ惑うユニット

ログラムは Python と機械学習ライブラリである Chainer[*5] を用いて作成した。制御プログラムと学習プログラムは、MessagePack-RPC[*6] を用いて通信を行った。

125 万イテレーションの学習を行った結果、**図 6** のように敵から逃げ惑うような動きが学習された。図中の黄色い線は、移動方向を示している。

学習に与えられる報酬は、敵に与えたダメージと敵に与えられたダメージの差であるため、敵から逃げることによって短期的な報酬を向上させることができるためと考えられる。しかし、戦闘に勝利するためには敵を倒す必要があるため、あまり良い動きとはいえない。

また、多くのユニットは、**図 7** のように初期位置からまず下に移動し、角を過ぎたあたりで左に曲がる行動をとった。このことから、各ユニットは自分のマップ上の位置を理解し、うまく逃げるように学習されたと考えられる。

少ない損害で多くのダメージを与えるための一般的な戦略として、素早く敵ユニットの数を減らす手法がある。それを達成するためのいくつかの方法がある。一つは全味方ユニットで 1 体の敵ユニットを集中砲火することである。

*5 http://chainer.org/
*6 https://github.com/msgpack-rpc

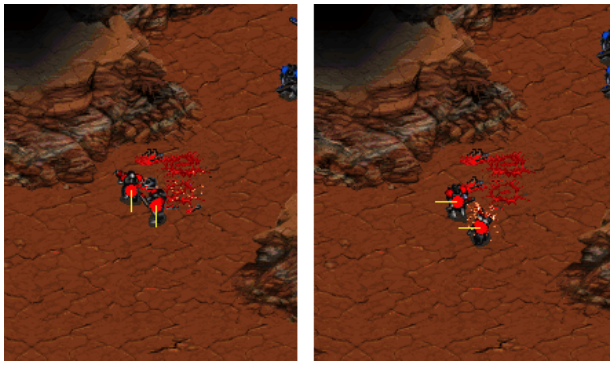


図 7 角を回りこむ様子

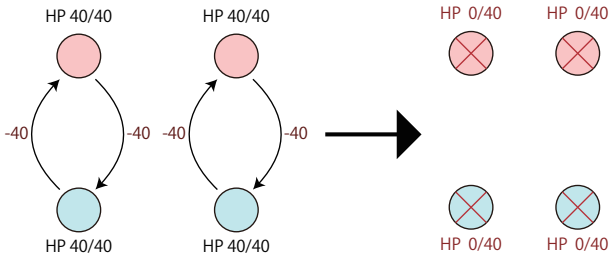


図 8 一対一に攻撃した場合

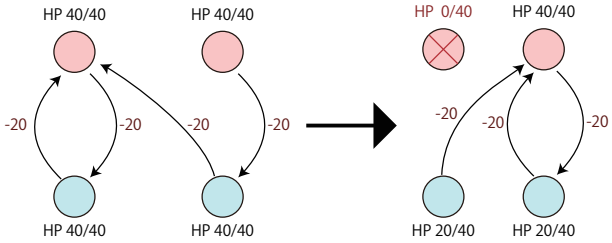


図 9 集中砲火した場合

図 8 のように 2 体ずつのユニットが一対一に攻撃した場合共倒れになってしまうが、**図 9** のように集中砲火をした場合は、1 ユニット生き残ることができる。ただし、敵の HP より多くのダメージを与えることは非効率なので、必要最小限のユニット数を割り当てる必要がある。もう一つの方法は、HP が低い敵ユニットを優先して攻撃することである。

その制御のためには、戦闘において大局的に盤面を判断し各ユニットに司令をだす上位の AI が必要である。しかし本手法では、各ユニットは個別の判断に基づき行動し、また攻撃対象には最も近い敵を選択する。そのため戦力がばらけてしまい、損害より多くのダメージを与えることができず、正の報酬が得られなかったと考えられる。各ユニットが個別に下す判断と戦闘における大局的な判断をうまく統合する必要があるが、本研究の提案手法にはそれが盛り込まれていない。大局的な判断と各ユニットの判断をうまく統合する手法に関する更なる検討が必要である。

また、ユニット同士で行動決定を共有していないため、**図 10** のように味方ユニットの移動方向が噛み合わず、お互いに引っかかって移動ができない様子も見られた。

図 10 ユニット同士で引っかかっている様子

今回の実験では、1 シチュエーションのみで学習・実験を行ったが、実際のゲームに適用するにはあらゆるシチュエーションに対応する行動を学習する必要がある。

6. おわりに

本論文では、StarCraft の戦闘に関して、DQN を用いてユニットコントロールを学習する手法を提案した。地形の情報をグリッドマップとして DQN に入力して学習することによって、周囲の地形の状態を加味しつつ敵ユニットから逃げるような AI が学習された。しかし、実際に戦闘に勝利するためには逃げるだけではなく、攻撃を行って敵ユニットを倒す必要がある。そのためには、戦闘における大局的な判断に基づき味方ユニット同士が協調し、注意深く攻撃対象を選ぶ必要がある。

今後の方針としては、学習方法 2 に対する実験を行い、また本実験で見つかった課題点に対する改善案を検討する。

参考文献

[1] 鎌田徹朗，橋本　剛，高野誠也：StarCraft AI への隊列導入，技術報告 10，松江工業高等専門学校 (2015).
[2] Tung, N., Kien, N. and Ruck, T.: Potential flow for unit positioning during combat in StarCraft, *IEEE 2nd Global Conference on Consumer Electronics (GCCE 2013)*, IEEE, pp. 10–11 (2013).
[3] Wender, S. and Watson, I.: Applying reinforcement learning to small scale combat in the real-time strategy game starcraft: broodwar, *IEEE Conference on Computational Inteligence and Games (CIG 2012),*, IEEE, pp. 402–408 (2012).
[4] Zhe W., Kien Quang N., Ruck T., Frank R.: MONTE-CARLO PLANNING FOR UNIT CONTROL IN STAR-CRAFT, *The 1st IEEE Global Conference on Consumer Electronics 2012*, pp. 263–264 (2012).
[5] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.: Playing Atari With Deep Reinforcement Learning, *NIPS Deep Learning Workshop* (2013).