自己対戦を用いたMini-RTSの均衡戦略の求解

河村 圭悟[1,a)]　鶴岡 慶雅[2]

概要：強化学習は Atari 2600 などのビデオゲームで多くの成功を収めているが， Real-Time Strategy (RTS) ゲームにおいては，強化学習を用いた AI はまだ人間のトッププロに勝利できていない．その理由の一つには， RTS ゲームが時間的に定常な 1 人ゲームであるとみなすことができず，強化学習の仮定を満たせないということが挙げられる．本研究では，研究用の RTS ゲームである ELF Mini-RTS を対象とし，一般の RTS ゲームに解を与えるための第一歩として， Mini-RTS を二人零和展開型ゲームであるとみなしてゲーム理論的なアプローチで実験的に解を求めた．より具体的には，ナッシュ均衡戦略を求める手法である Neural Fictitious Self-Play (NFSP) が方策勾配法とも組み合わせられることを示し，これを Mini-RTS に適用することで戦略を計算した．実験の結果，方策勾配法を組み合わせた NFSP は Mini-RTS における搾取されにくい戦略を求めることができ，かつその学習速度は通常の自己対戦を用いて事前学習することで向上させられることが示された．

Computing Equilibrium Strategies for Mini-RTS with Self-Play

Keigo Kawamura[1,a)]　Yoshimasa Tsuruoka[2]

Abstract: Despite the notable successes in video games such as Atari 2600, current AI is yet to defeat human champions in the domain of Real-Time Strategy (RTS) games. One of the main reasons is that an RTS game is a multi-agent game, in which reinforcement learning methods cannot simply be applied because the environment is not a stationary Markov decision process. In this paper, we present a first step toward finding a game-theoretic solution to RTS games by using Mini-RTS, a small but nontrivial RTS game provided on the ELF platform, as our testbed. More specifically, we show that Neural Fictitious Self Play (NFSP), a game-theoretic approach for finding Nash equilibria, can be effectively combined with policy gradient reinforcement learning and be applied to Mini-RTS. Experimental results also show that the scalability of NFSP can be substantially improved by pretraining the models with simple self-play using policy gradients, which by itself gives a strong strategy despite its lack of theoretical guarantee of convergence.

1. はじめに

近年の深層学習の発展によって，強化学習は多くの複雑なゲーム環境で成功を収めている．特に，Atari 2600 のゲーム環境では，強化学習によって画像情報のみからほとんどすべてのゲームについて人間を超えるスコアが達成されている [1], [2], [3]．しかしながら，リアルタイムストラテジー (RTS) ゲームの分野では，強化学習を用いた AI はまだ人間のトッププレイヤの実力を上回っていない [4]．RTS ゲームは，チェスや囲碁に続く多人数強化学習における次世代の挑戦対象であると考えられており [5], [6]，この RTS ゲームにおいて強いプレイヤを求める手法が望まれている．

大規模な RTS ゲームにおける研究を効率よく行うために，いくつかの研究用ゲーム環境が提案されている [4], [7], [8]．強化学習の研究を高速に行うためのプラットフォームの一つである ELF (Extensive, Lightweight, and Flexible platform) [5] では， Mini-RTS と呼ばれる研究用の RTS ゲー

[1] 東京大学大学院工学系研究科電気系工学専攻
Department of Electrical Engineering and Information Systems, Graduate School of Engineering, The University of Tokyo
[2] 東京大学大学院情報理工学系研究科電子情報学専攻
Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo
[a)] kkawamura@logos.t.u-tokyo.ac.jp

ムが提供されている．このゲームは RTS ゲームとしては小規模であるものの， fog-of-war や リソース管理など RTS ゲームの特徴をすべて保持しており，かつ既存の他のゲーム環境と比較して高速に動作する．本研究では，より大規模な RTS ゲームを解くための第一歩として，この Mini-RTS におけるゲーム理論的な意味での解を求めることを目指す．

RTS ゲームにおける強いプレイヤを作成するためには，戦略と戦術のトレードオフ，リアルタイムな意思決定，事前知識の活用など，様々な難点が存在している [9], [10] が，本研究ではそのうち多人数性を主な研究対象とする．RTS ゲームは二人以上の多人数ゲームであり，学習するプレイヤにとって環境が時間的に定常ではない．これは，強化学習が環境に対して設ける仮定の一つである Markov Decision Process (MDP) 性に反する．

多人数ゲームで強化学習を行う手法として，自己対戦 (self-play) が挙げられる [6], [11]．最も単純な自己対戦では，各プレイヤが同時に相手プレイヤに対する報酬を最大化するように学習する．自己対戦では，各プレイヤが合理的に (rational) 学習していて，かつその学習が収束すれば，その戦略の組はナッシュ均衡になることが知られている [12]．しかしながら，自己対戦では学習が収束する保証はなく，実際に収束せず振動する例も観測されている [13]．

Heinrich らは，これらの手法とは根本的に異なるアプローチとして，ゲーム理論の Fictitious Play (FP) に相当するアルゴリズムを機械学習を用いて行う Fictitious Self-Play (FSP) と呼ばれる手法を提案した [14]．FP はいくつかの条件の元でナッシュ均衡戦略に収束することが証明されているため，その近似である FSP も通常の自己対戦と比べてより収束する可能性が高いと考えられる．実際に， FSP に深層学習の手法といくつかの改良を加えた NFSP (Neural Fictitious Self-Play) は，単純なポーカーのゲーム環境において，事前知識を用いることなく近似的なナッシュ均衡戦略を得ることに成功した [15]．

本研究では，この NFSP に対して方策勾配法 (policy gradient method) が適用でき，かつその手法を用いて Mini-RTS の均衡解を得ることができることを示す．また，方策勾配法と組み合わせることによって， NFSP のネットワークを通常の self-play の学習で事前学習できるようになり，それによって NFSP の学習速度を向上させられることを示す．この研究は，我々の知る限り初めて非自明な RTS ゲームの均衡解を求めることに成功した研究であり，したがってより複雑な RTS ゲームを解くための重要な第一歩である．

2. 背景

2.1 強化学習と Markov Decision Process

強化学習では，環境のモデルとして MDP を仮定することが多い． MDP 環境 $\mathcal{E}$ における強化学習では，プレイヤは各時刻 t で状態 $s_t \in \mathcal{S}$ を受け取り，合法手集合 $\mathcal{A}$ 上の確率分布を表す関数 $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ から行動 $a_t \in \mathcal{A}$ を選択する．この関数を方策 (policy)，あるいは多人数ゲームの文脈では戦略 (strategy) と呼ぶ．環境 $\mathcal{E}$ は選択された行動 a_t を実行し，状態遷移関数 $T(s_{t+1} \mid s_t, a_t)$ と報酬関数 $R(r_{t+1} \mid s_t, a_t, s_{t+1})$ を用いて次状態 s_{t+1} と報酬 r_{t+1} を返す．プレイヤの目的は，累積期待報酬 $\mathbb{E}[R_t] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right]$ を最大化することである．

あるプレイヤから見て区別できないような状態がある場合，その MDP 環境は Partially Observable MDP (POMDP) と呼ばれる．POMDP 環境 $\mathcal{E}_p$ における強化学習では，プレイヤは観測 $o_t = O(s_t)$ を受け取り，行動 $a_t \in \mathcal{A}$ を戦略からサンプルすることで選択する．観測関数 O は，環境 $\mathcal{E}_p$ の真の状態 s からプレイヤへの観測を与える関数であり，プレイヤの戦略 π は状態ではなく観測に依存する．

MDP 環境の状態 s について， s での累積期待報酬を表す状態価値関数 $V_\pi(s)$ ， s から行動 a を取った後の累積期待報酬を表す行動価値関数 $Q_\pi(s,a)$ ， V に対する Q の利得差を表す advantage $A_\pi(s,a)$ を，以下で定義する．

$$Q_\pi(s_t, a_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right]$$

$$V_\pi(s_t) = \sum_{a \in \mathcal{A}} \pi(a \mid s_t)\, Q_\pi(s_t, a)$$

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$$

2.2 展開型ゲーム

本研究では， RTS ゲームを展開型ゲームとして扱うことで均衡戦略の計算を行う．展開型ゲームは多人数の不完全情報ゲーム環境を表すモデルの一つであり，各プレイヤから見た状態をゲーム木のように扱うことで状態遷移を表現する．

展開型ゲームにおいては，各プレイヤが必ずしも状態のすべてを観測できるわけではない．プレイヤ i から見て区別できないような状態の集合を情報集合と呼び， $I_i(s)$ で表す．各プレイヤは，各情報集合に対する行動の確率分布である戦略 π_i を持つ．また，すべてのプレイヤの戦略の組 $\pi = \{\pi_1, \cdots, \pi_N\}$ を戦略プロファイルと呼ぶ．戦略プロファイルに対する各プレイヤの期待累積報酬を $R_i(\pi)$ と表記する．あるプレイヤ i について，i 以外の戦略 $\pi_{-i} = \{\pi_j \mid j \in \mathcal{N}, j \neq i\}$ を固定し，その戦略に対する期待累積報酬を最大化する戦略 $\pi_i^* = \arg\max_{\pi_i} [R_i(\pi_i, \pi_{-i})]$ を考えるとき，この戦略を π_{-i} に対する最適応答戦略と呼ぶ．すべてのプレイヤの戦略が，自分以外の相手に対して最適応答戦略になっているような戦略プロファイルを，

ナッシュ均衡戦略と呼ぶ．

プレイヤが 2 人であり，かつ常に報酬の和が 0 であるとき，このゲームは二人零和ゲームであるという．また，各プレイヤが一度観測した状態をすべて記憶しているとき，このゲームは完全記憶 (perfect recall) ゲームであるという．

展開型ゲームは，あるプレイヤに注目し，その他のプレイヤの戦略を固定することで，POMDP であるとみなすことができる．さらに，そのゲームが完全記憶ゲームであるとき，このゲームは MDP であるとみなすことができる．

2.3 Neural Fictitious Self-Play

Neural Fictitious Self-Play (NFSP) は， FSP の関数近似器として，ニューラルネットワークと Deep Q-Networks (DQN) [1] を用いたものである．FSP は，標準型ゲームにおけるナッシュ均衡戦略を求める手法である Fictitious Play (FP) を展開型ゲームにも適用できるようにし，さらに機械学習の手法を用いてテーブルを持てないようなゲームにも適用できるようにしたものである．

FP では，繰り返し self-play を行いつつ，各イテレーションで相手プレイヤの平均戦略に対する最適応答戦略を計算する．同時に，各プレイヤはこれまでの最適応答戦略の平均となる戦略を計算する．FSP では，この計算のうち最適応答戦略を求める部分を強化学習で，平均戦略を求める部分を教師あり学習で近似することで，すべての情報集合における確率分布を保持・更新しなければならないという問題点を解消しつつ，最新の機械学習のアルゴリズムを適用できるようにしている．

FSP は特定の機械学習の手法に依存しない汎用的なアルゴリズムであるが，ここにニューラルネットワークによる強化学習・教師あり学習を適用したのが NFSP である，また，NFSP では教師あり学習のサンプリング方法を改善するために，通常の circular replay buffer ではなく reservoir buffer を用いている．これによって，有限のメモリでもデータ全体から等確率でサンプリングして保持することができ，平均戦略の計算を効率よく行えるようになっている．また， NFSP では教師あり学習が相手の変化により鋭敏に反応できるようにするために， anticipatory dynamics [16] を用いて buffer 内のデータの先読みを行っている．

2.4 Proximal Policy Optimization

DQN のような value-based な手法は，確率的なプレイヤを表現できないため，後述する通り通常の自己対戦ではナッシュ均衡解が存在しない可能性がある．NFSP に適用することで間接的に確率的なプレイヤを表現することはできるが，本研究ではより直接的に， policy-based な手法である Proximal Policy Optimization (PPO) [17] を強化学習として用いた．

PPO は, Trust Region Policy Optimization (TRPO) [18] を単純化し，性能を落とすことなく実装面・速度面で改善させた手法である．TRPO では， surrogate objective function と呼ばれる目的関数を，ある条件の元で最大化することで戦略の更新を行う．

この最適化問題における制約条件は，確率分布がデータ生成時の確率分布から外れすぎて，データに対する更新が真のパラメータを悪化させるのを防ぐために導入されている． PPO では，この制約条件を単に確率のクリッピングに置き換えている．PPO では以下の目的関数を制約条件なしに最大化する．

$$L(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t \hat{A}_t,\, \mathrm{clip}\left(r_t,\, 1-\epsilon,\, 1+\epsilon\right) \hat{A}_t \right) \right] \quad (1)$$

ここで， $r_t = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}$ は現在の確率の元の確率に対する比であり， ϵ は比の上限を決めるしきい値， $\hat{A}_t$ は advantage A_t の推定量である．この最適化問題は TRPO の最適化問題に比べて単純であり，かつ実験的に TRPO よりよい性能が出ることが知られている．

3. ELF Mini-RTS

本研究の目的は， ELF Mini-RTS において，任意のプレイヤに搾取されないような均衡戦略を求めることである．

Mini-RTS におけるプレイヤの目的は，戦車を作成し，相手の基地を破壊することである．各プレイヤは自分の基地，数種類のユニット，そしてリソースを持っている．プレイヤはリソースを消費することで兵士を作成でき，空き地に兵士を送りリソースを消費することで兵舎を作成でき，兵舎に兵士を送りリソースを消費することで戦車を作ることができる．

Mini-RTS はフレーム単位で動作する．プレイヤは各フレームで各ユニットにどの行動を行うかの命令を送り，それらの行動が実行されて環境が遷移し，新たな観測が返る，というサイクルを繰り返すことでゲームが進行する．ゲームには fog-of-war があり，観測は自分のユニットがいる周囲についてのみ観測可能であり，他の部分はマスクが掛けられている．したがって，このゲームは不完全情報ゲームである．

「兵士 1 を 2 ピクセル右に移動させる」といった局所的な命令を指定する代わりに， Mini-RTS では「空いている兵士に兵舎を建設させる」「基地を防衛する」といった大域的な命令を選択することができる．これらの命令は全ユニットに同時に作用する．簡単のため，本研究ではこれらの大域的な命令のみを用いた．具体的には，プレイヤは各フレームで 9 つの行動から 1 つを選択する．その内訳は，建設に関する行動が 4 種類（兵士・兵舎・近接戦車・遠隔戦車），攻撃に関する行動が 4 種類（攻撃，離れて攻撃，攻撃して戻る，基地を防衛する），何もしない行動が 1 種類と

なっている．

また，環境からプレイヤに与えられる観測は，ゲーム画面を 20×20 のグリッドに分割し，その各グリッドにあるオブジェクトや基地の残り耐久力などを表す $22 \times 20 \times 20$ の値である．

4. 提案手法

本研究では，RTS ゲームを二人零和完全記憶展開型ゲームであるとみなし，自己対戦の手法を適用することで均衡戦略を得ることを目指す． RTS ゲームには多くのユニットが存在しているが，プレイヤはすべてのユニットの観測を得ることができ，かつすべてのユニットに命令を出すことができる．したがって，プレイヤが二人であれば，勝敗のみを報酬とすることでこのゲームは二人零和ゲームとみなすことができる．また，先述した通り， fog-of-war によって観測が制限されているため，このゲームは不完全情報ゲームである．ほとんどの RTS ゲームは離散的なフレームで管理されており，プレイヤの意思決定が十分速ければこれは順番に手番を行うゲームであるとみなせるため，このゲームは展開型ゲームであるとしてもよい．ゲームが完全記憶であるかどうかは，プレイヤが過去の観測をすべて記憶しているかどうかに依存する．このことについては第 6 節で後述する．

完全情報ゲームでは，ナッシュ均衡であるような戦略プロファイルの中に，すべての状態において確定的に行動を決定するような戦略プロファイルがあることが知られている．しかしながら，不完全情報ゲームにおいては，そのような戦略プロファイルは一般には存在しない．確定的でない（確率的な）プレイヤを通常の自己対戦でも得るために，本研究では NFSP の強化学習アルゴリズムとして，方策勾配法の一種である PPO を用いた．この NFSP と PPO を組み合わせたアルゴリズムをアルゴリズム 1 に示す．

このアルゴリズムでは，行動を決定する関数と学習を行う関数がコールバックとして各 Γ_i に登録される．10 行目の各ステップにおいて，各 Γ_i では複数のゲームスレッドが同時に実行され，いずれかのコールバックが適切なバッチとともに呼び出される．

本研究では， ゲームインスタンス Γ_i を各プレイヤ p について並列に作成し，実行した．各インスタンスではプレイヤ p が強化学習に，それ以外のプレイヤが教師あり学習に従って行動し，各 Γ_p ではプレイヤ p のみが学習を行うようにした．このアルゴリズムは，各ゲームの先頭で強化学習か教師あり学習を選択し，各プレイヤを等価に扱うという元の NFSP のアルゴリズムとは大きく異なっている．元のアルゴリズムでは，各プレイヤ p は 4 種類の経験データを得ることができる．すなわち，(π_p, π_{-p}) ，(π_p, β_{-p}) ，(β_p, π_{-p}) ，(β_p, β_{-p}) である．ここで， π は教師あり学習を，β は強化学習を表す．また， $-p$ はプレイヤ p を除く全プレイヤを表す．もし強化学習が off-policy であれば，自身が β_p で行動したデータだけでなく， π_p に従って行動したデータも強化学習に用いることができる．しかしながら，もし強化学習が on-policy であれば， $(\pi_p, \cdot_{-p})$ は強化学習の学習には用いることができなくなってしまう．教師あり学習も自身が π_p に従って行動したデータは学習に用いないので，強化学習が on-policy の場合，元のアルゴリズムのままでは多くのデータが無駄になってしまう．これを避けるため，アルゴリズム 1 では学習するプレイヤのみが強化学習に従って行動を行うようになっている．

アルゴリズム 1 NFSP with PPO

$\Gamma = \{\Gamma_i \,|\, i = 1, 2, \cdots, N\}$ is a set of Game instances and N is the number of players
1: **function** MAIN(Γ, N)
2: **for** $p = 1, 2, \cdots, N$ in parallel **do** ▷ p is a learning player
3: Γ_p.Initialize()
4: Γ_p.RegisterCallback(p, Trainer)
5: Γ_p.RegisterCallback(p, RLActor)
6: **for** $q = 1, \cdots, p-1, p+1, \cdots, N$ **do**
7: Γ_p.RegisterCallback(q, SLActor)
8: **end for**
9: **repeat**
10: Γ_p.DoStep() ▷ Multiple games are executed concurrently and a corresponding callback is called at a step
11: **until** Time steps exceed the certain limits
12: **end for**
13: **end function**
14: **function** TRAINER(p, batch)
15: $\{S_\tau, A_\tau, \Pi_\tau, R_\tau\}_{\{\tau=t,\cdots,t+T-1\}} \leftarrow$ batch
▷ State, action, probability distribution, and reward
▷ Π_t is a probability distribution of NN_{RL} at t
16: Calculate $\mathcal{L}_{RL}$ with eq. (2)
17: Memorize $\{S_\tau, \Pi_\tau\}$ in buffer $\mathcal{M}_{SL}$
18: Sample $S, \Pi \leftarrow \mathcal{M}_{SL}$
19: Calculate $\mathcal{L}_{SL}$ with closs entropy
20: Optimize NN_{RL} and NN_{SL} with $\mathcal{L}_{RL}$ and $\mathcal{L}_{SL}$
21: **end function**
22: **function** RLACTOR(p, batch)
23: $S \leftarrow$ batch
24: $\pi \leftarrow \mathrm{NN}_{RL}(S)$
25: **return** Sampled $a \leftarrow \pi$
26: **end function**
27: **function** SLACTOR(p, batch)
28: $S \leftarrow$ batch
29: $\pi \leftarrow \mathrm{NN}_{SL}(S)$
30: **return** Sampled $a \leftarrow \pi$
31: **end function**

16 行目において， $\mathcal{L}_{RL}$ は PPO と同様に，以下の式で計算される．

$$\mathcal{L}_{RL} = \mathcal{L}_{policy} + \alpha \mathcal{L}_{entropy} + \beta \mathcal{L}_{value} \qquad (2)$$

ここで， $\mathcal{L}_{policy}$ は式 (1) で表される PPO の目的関数を -1 倍したものであり，$\mathcal{L}_{entropy} = \sum_a \pi_\theta(a \,|\, s) \log \pi_\theta(a \,|\, s)$ はプレイヤに探索を行わせるための正則化項である．ま

た，$\mathcal{L}_{value}$ は V_θ と V_{target} のクリッピング付き二乗誤差である．具体的には，式 (1) において，推定量 $\hat{A}$ は

$$\delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)$$
$$A_t = \delta_t + k\delta_{t+1} + \cdots + k^{T-t+1}\delta_{T-1}$$
$$\bar{A}_t = \frac{A_t - \mathrm{mean}_{t'}(A_{t'})}{\mathrm{stdev}_{t'}(A_{t'}) + \epsilon_{std}}$$

のように計算され，$\mathcal{L}_{value}$ は

$$V_{clip}(s) = \mathrm{clip}\left(V_\theta(s) - V_{\theta_{old}}(s), -\epsilon_v, \epsilon_v\right) + V_{\theta_{old}}(s)$$
$$\mathcal{L}_{value} = \max\left((V_\theta) - V_{target})^2, (V_{clip} - V_{target})^2\right)$$

と計算される．

本研究では，V_{target} として $V_{target} = A_t + V_{\theta_{old}}$ を用いた．これは OpenAI Baselines [19] の実装と同じである．

NFSP に加え，本研究では PPO を用いた通常の自己対戦も使用した．アルゴリズムは NFSP のものとほとんど同一であるが，教師あり学習が存在せず，すべてのプレイヤが強化学習に従って行動する点が NFSP とは異なっている点である．

5. 実験

5.1 実験設定

実験はすべて，以下の設定で行った．

バッチサイズは 128 とし，時間方向のバッチサイズは 50 とした．アルゴリズム 1 の 18 行目の reservoir sampling では，一度に 512 個のデータをサンプルした．フレームスキップは 50 に設定した．すなわち，各プレイヤは 50 フレームごとに行動を選択するようにした．各 Γ_i では 512 ゲームを並列に実行した．ニューラルネットワークのモデルとしては，64 チャンネル，3×3 カーネルの畳み込み層と $\alpha = 0.1$ の leaky ReLU を組み合わせたレイヤを 4 枚並べ，2 枚ごとに 2×2 max pooling レイヤを挟むことで body block を構成した．すべてのモデルはこの body block に適切なヘッダ π_{SL}，π_{RL}，または V_{RL} をつけた構造になっており，ヘッダはすべて 1 層の全結合層からなっている．ただし，$\pi_\cdot$ には確率分布を表すためにソフトマックス層が付いている．Body block のパラメータは，π_{RL} と V_{RL} を共有させ，π_{SL} は別のパラメータを用いた．また，プレイヤ間ではすべてのネットワークを共有した．最適化には勾配を 0.5 でクリッピングした確率的勾配降下法を用いた．学習率として強化学習のモデルには 0.01 を，教師あり学習のモデルには 0.001 を用いた．式 (2) の計算に用いるハイパーパラメータは，$\alpha = 0.01$，$\beta = 0.5$，$\gamma = 0.99$，$k = 0.95$，$\epsilon_{std} = 10^{-8}$，$\epsilon_v = 0.1$ を用いた．

5.2 Mini-RTS における自己対戦

通常の自己対戦および NFSP を用いて Mini-RTS の均衡解が得られるかを確認するために，これらの手法を用いてプレイヤを学習させ，ルールベース AI に対する勝率を計測する実験を行った．Mini-RTS には AI-Simple と AI-Hit-and-Run の 2 種類のルールベース AI が用意されており，前者は単にユニットを一定数生産して攻撃するだけだが，後者はより複雑な戦略を用いるようになっている．人間のプレイヤにプレイさせた結果，それぞれに対して 90%，50% の勝率であったことが報告されている [5]．

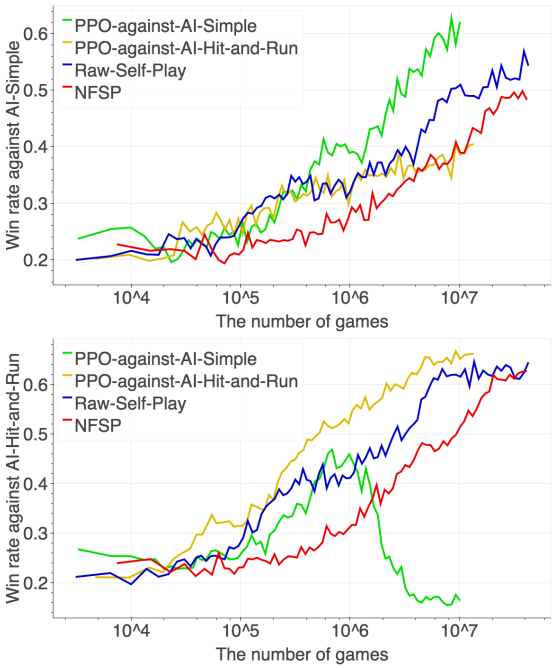

図 1 Mini-RTS における各プレイヤの勝率．横軸は学習に用いたゲームの数を対数で示している．上図が AI-Simple との対戦結果を，下図が AI-Hit-and-Run との対戦結果を示している．

このゲームは二人零和対称ゲームなので，もしプレイヤがナッシュ均衡戦略の一部であるような戦略を取っていれば，そのプレイヤはどんなプレイヤにも搾取されず，必ず 50% 以上の勝率を得る．ゲームが十分小さければ，戦略がどれだけナッシュ均衡戦略に近いかを表す可搾取量を計算することができる [20] が，Mini-RTS は状態量が多すぎるため，このような評価手法は適用できない．可搾取量の近似や下界を与えるアルゴリズムは他にも存在するが [21]，上界を計算することはできず，Mini-RTS のような十分大きいゲームで戦略がナッシュ均衡に近づいていることを確実に評価する手法は存在しない．本研究では，単に評価対象となる戦略をルールベース AI と対戦させ，その勝率を評価指標とした．この値は可搾取量の下界の推定量になっている．

図 1 に実験の結果を示す．AI-Simple に対する勝率で比較すると，対戦相手を AI-Simple に固定して学習させた PPO プレイヤが最も高い勝率を得ているが，このプレイヤは AI-Hit-and-Run に対する勝率が非常に低くなっており，学習時の対戦相手に特化した戦略を学習したことで搾取されやすい戦略になってしまっていることがわかる．

AI-Hit-and-Run に対して学習させた PPO プレイヤも同様に，学習時の相手に対しては最も良い勝率を出すことができるが，それ以外の相手には搾取されてしまうことがわかる．

NFSP を用いて学習させたプレイヤは，学習は他の手法と比べて遅いものの，どちらのルールベース AI に対する勝率も学習が進むにつれて上がっていることがわかる．このことから，単純な PPO のような特定の相手に特化した戦略ではなく，より搾取されにくい，ナッシュ均衡に近い戦略が得られていると考えられる．

また，通常の自己対戦を用いて学習させたプレイヤは，AI-Hit-and-Run に対する勝率で NFSP より早く同じ結果に到達しており，AI-Simple に対する勝率では NFSP よりも高い勝率を得ることに成功している．NFSP は最適応答戦略と平均戦略の両方を同時に学習する必要があるのに対して，通常の自己対戦では最適応答戦略のみを学習すれば良いので，もし通常の自己対戦もナッシュ均衡戦略に収束するのであれば NFSP よりも速く収束することが期待される．通常の自己対戦では戦略が収束する理論的な保証はないが，戦略が収束すればその戦略はナッシュ均衡であるため，実験的には NFSP よりも良くなることがある．実際に自己対戦が振動して収束しない例も報告されている [13] が，今回の実験ではそのような結果は観測されなかった．

自己対戦でない通常の強化学習アルゴリズムを用いて学習させる際に，Tian らは AI-Simple と AI-Hit-and-Run を 50% ずつ混ぜ合わせた AI を対戦相手として学習させることで性能が向上することを示した [5] が，今回はこの実験は行わなかった．本論文の目的は任意の相手に対して搾取されないプレイヤを構成することであり，そのためには少なくとも 1 つ学習時に見ることのできない対戦相手を用意する必要があるからである．

図 1 から分かる通り， PPO を用いて特化した戦略を学習させた場合でも，勝率は 65% 程度に留まっている．これは， Mini-RTS がランダム要素の強いゲームであることに起因する．Mini-RTS では，ゲーム開始時に，リソースや基地の位置，ユニットなどがお互いのプレイヤにランダムに配られる．このユニットの数などは非対称であり，かつフレームスキップが大きいため，もしゲーム開始時に相手プレイヤが多数のユニットを持っており，自分が一切ユニットを持っていなかった場合，プレイヤは相手プレイヤの攻撃をどうやっても防ぐことができない．実際，純粋なランダムプレイヤに対して PPO を用いて 10^7 ゲーム学習させたプレイヤを用意し，簡易的な実験を行ったところ，同じランダムプレイヤに対する勝率は 71% に留まってしまい，十分学習させたプレイヤであっても 29% のゲームでランダムプレイヤに敗北してしまうという結果が得られた．

実験で得られたプレイヤをより詳細に分析するため，別の PPO プレイヤを用意し，実験で得られたプレイヤに対して 10^7 ゲーム対戦させて学習させ，勝率を計測した．もし PPO が最適な戦略に収束すれば，この勝率は対象となるプレイヤの，不完全記憶かつフレームスキップがある状況下における可搾取量を表すことになる．

表 1 各プレイヤに対して PPO を用いて最適応答戦略を計算させたときの PPO プレイヤの勝率．† が付いている結果を除いて，いずれも 10^7 ゲームの学習を行った．

Agent	Win rate
Random	0.71
AI-Simple	0.62
AI-Hit-and-Run	0.65
PPO against AI-Simple	$> 0.80^{\dagger}$
PPO against AI-Hit-and-Run	0.56
Raw self-play with PPO	**0.45**
NFSP with PPO	**0.44**

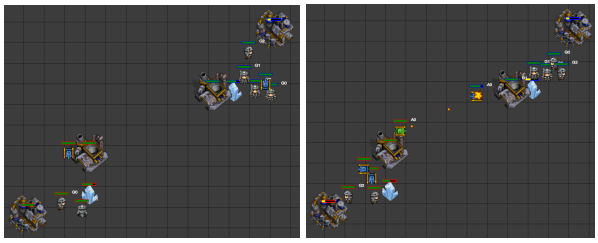

図 2 NFSP プレイヤ（左下赤枠）と，それに対して学習させた PPO プレイヤ（右上青枠）の対戦のスクリーンショット．青色の戦車は近接戦車を，緑色の戦車は遠隔戦車を表す．NFSP プレイヤは（画像左）まず近接戦車を建設し，（画像右）次に遠隔戦車を建設した．

結果を表 1 に示す．他のプレイヤと比較して，自己対戦及び NFSP を用いて得られたプレイヤが搾取されにくい戦略になっており， PPO が 50% 未満しか搾取できていないことがわかる．この結果は，これらのアルゴリズムによって得られた戦略が，不完全記憶かつフレームスキップがあるような任意の戦略に対して搾取されないことを示唆している．

得られた戦略が具体的にどのようなものなのかを確かめるため， NFSP によって得られたプレイヤとこれに対して学習させた PPO プレイヤの対戦結果のスクリーンショットを図 2 に示した．この対戦では， NFSP プレイヤはまず 2 体の近接戦車を建設し，1 体の遠隔戦車を建設し，その後に一斉攻撃を仕掛けている．この行動は人間にとって合理的な挙動である．なぜなら，ゲームの性質上，近接・遠隔戦車はそれぞれ防御・攻撃に優れており， fog-of-war によって相手がいつ攻撃してくるかわからない状況下では，まず防御ユニットを建設してから攻撃ユニットを建設することで相手の攻撃に備えつつ攻撃の準備を行うことができるからである．

図 3 に別の例を示す．この例では， Mini-RTS のランダム性により， ゲーム開始時点で PPO プレイヤには兵舎が

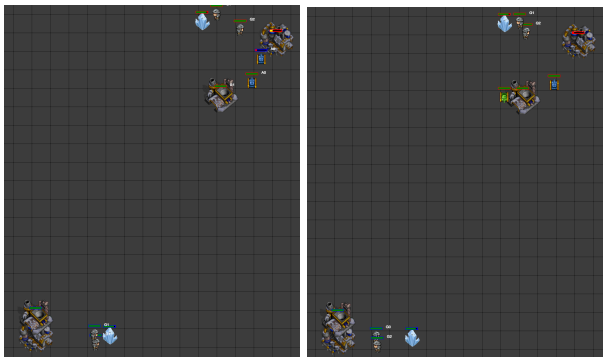

図 3　NFSP プレイヤ（右上赤枠）と，それに対して学習させた PPO プレイヤ（左下青枠）の対戦のスクリーンショット．（画像左）PPO プレイヤの戦車によって NFSP プレイヤの基地が攻撃されているが，かろうじて耐えている．（画像右）相手の攻撃を耐えた直後，NFSP プレイヤは遠隔戦車を建設した．

ある一方で，NFSP プレイヤには兵舎がないという大きな差が開いている．NFSP プレイヤが兵舎を建設している間に，PPO プレイヤは兵舎を使って戦車を建設し，NFSP プレイヤに攻撃を仕掛けてきている．しかし，NFSP プレイヤは防御に優れた近接戦車ではなく，攻撃に優れた遠隔戦車を新たに建設し，相手プレイヤにカウンターを仕掛けることでこの試合に勝利している．今回の実験設定ではすべての命令が大域的であるため，相手が攻撃を仕掛けてきた時点で相手は他に防衛のための戦力を残していないことがわかる．NFSP プレイヤはこのことを利用してカウンターを成功させている．

5.3　自己対戦による NFSP の事前学習

前節において，NFSP は搾取されにくい戦略を得られるものの，他の手法に比べて学習が遅いという問題点が観測された．これに対して，PPO を用いた通常の自己対戦は，速く収束し特定の相手に依存しない戦略が得られるものの，収束する理論的保証がないという問題点があった．もし自己対戦で学習したプレイヤを用いて NFSP プレイヤの事前学習ができれば，双方の問題点を打ち消すことができると考えられる．

まず，NFSP の強化学習のパラメータを通常の自己対戦によって事前学習させ，勝率を計測した．結果を図 4 に水色の線で示した．学習は事前学習を行わない NFSP と比べて多少速くなっているものの，NFSP プレイヤの性能を大きく向上させることはなかった．これは，NFSP が教師あり学習と強化学習の両方を同時に学習する必要があることに起因すると考えられる．NFSP の強化学習部分は自己対戦によって事前学習されているため，学習初期の段階で強化学習はある程度「良い」戦略を持っている．しかし，教師あり学習はランダムに初期化されているため，教師あり学習が強化学習の確率分布を学習するまでは強化学習はランダムプレイヤと対戦することになり，その間に事前学習が忘却されてしまう．これによって結果が向上しなかったのではないかと考えられる．

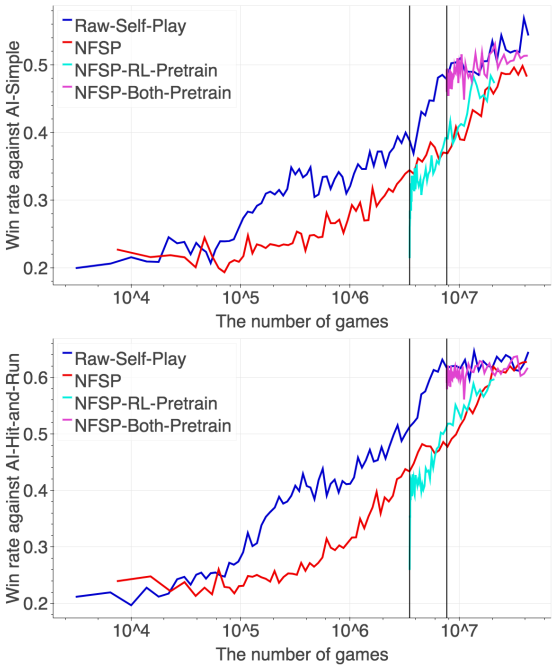

図 4　事前学習を用いた NFSP プレイヤの勝率．横軸は学習に用いたゲームの数を対数で示している．上図が AI-Simple との対戦結果を，下図が AI-Hit-and-Run との対戦結果を示している．黒線は事前学習から NFSP の学習に切り替えたタイミングを表している．

この問題を解決するため，自己対戦で学習したパラメータを，強化学習だけでなく教師あり学習のパラメータにも転用した．教師あり学習の確率分布を出力するモデルと強化学習の確率分布を出力するモデルは同一であるため，この操作は単に π_{RL} のパラメータを π_{SL} にコピーするだけで行うことができる．この実験の結果を，図 4 に紫色の線で示す．通常の自己対戦と比べると結果は下がっているものの，NFSP は事前学習した結果を崩壊させることなく維持し，強化学習のみを事前学習させた場合と比べても性能を向上させることに成功している．この結果は，学習は速いが収束の保証はない自己対戦を用いて，学習は遅いが収束を保証する理論的背景がある NFSP を事前学習させられることを示唆している．

6.　おわりに

本研究では，研究用の RTS ゲームである Mini-RTS に対して，これを二人零和展開型ゲームであるとみなして自己対戦と NFSP を適用することで，通常の強化学習の手法で得られたプレイヤよりも搾取されにくい均衡戦略を得ることに成功した．また，得られたプレイヤの挙動を観察し，人間の目から見て合理的な行動をしていることを確認した．

本研究の貢献として，方策勾配法のアルゴリズムを NFSP

と組み合わせることができ，かつそれが Mini-RTS に適用可能であることを示した点が挙げられる．このことは，より複雑で大規模な RTS ゲームを強化学習を用いて解くための重要な第一歩であると考えられる．また，方策勾配法を用いることによって，収束の保証がない自己対戦を用いて事前学習を行うことで，NFSP の学習の遅さを補うことができることを示したことも貢献として挙げられる．この手法は自己対戦が収束しないような状況下であっても，自己対戦を途中まで行って NFSP に切り替えることで効率よく均衡戦略を求めることができる可能性を示している．

第 4 節で述べたとおり，本論文では本来保持すべき過去の観測を記憶する構造をプレイヤに持たせなかった．これによってゲームのクラスは不完全記憶ゲームになってしまい， FSP が要求するゲームに対する仮定を満たすことはできていない．この問題を解決するためには，例えば deep recurrent Q-learning [22] などのように，強化学習のコントローラに long short-term memory などの記憶素子を付加することが考えられる．しかしながら，ゲームが完全記憶であることを担保するためには，同様の記憶構造を教師あり学習にも適用しなければならない．このことは教師あり学習の replay buffer のサイズを非常に小さくしてしまい，手法のスケーラビリティが失われてしまう．この問題を解決する手法を考案し，その効果を確かめることを今後の課題としたい．

参考文献

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D.: Human-level control through deep reinforcement learning, *Nature*, Vol. 518, pp. 529–533 (2015).

[2] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K.: Asynchronous Methods for Deep Reinforcement Learning, *ICML*, pp. 1928–1937 (2016).

[3] Van Seijen, H., Fatemi, M., Romoff, J., Laroche, R., Barnes, T. and Tsang, J.: Hybrid Reward Architecture for Reinforcement Learning, *NIPS*, pp. 5392–5402 (2017).

[4] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T. P., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J. and Tsing, R.: StarCraft II: A New Challenge for Reinforcement Learning, *arXiv:1708.04782* (2017).

[5] Tian, Y., Gong, Q., Shang, W., Wu, Y. and Zitnick, C. L.: ELF: An Extensive, Lightweight and Flexible Research Platform for Real-time Strategy Games, *NIPS*, pp. 2659–2669 (2017).

[6] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm, *arXiv:1712.01815* (2017).

[7] Ontanon, S.: The combinatorial Multi-armed Bandit problem and its application to real-time strategy games, *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 58–64 (2013).

[8] Synnaeve, G., Nardelli, N., Auvolat, A., Chintala, S., Lacroix, T., Lin, Z., Richoux, F. and Usunier, N.: TorchCraft: a Library for Machine Learning Research on Real-Time Strategy Games, *arXiv:1611.00625* (2016).

[9] Ontan, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D. and Preuss, M.: A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft, *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 5, No. 4, pp. 293–311 (online), DOI: 10.1109/TCIAIG.2013.2286295 (2013).

[10] Robertson, G. and Watson, I.: A Review of Real-Time Strategy Game AI, *Ai Magazine*, Vol. 35, pp. 75–104 (2014).

[11] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J. and Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning, *PLOS ONE*, Vol. 12, No. 4, pp. 1–15 (2017).

[12] Bowling, M. and Veloso, M.: Rational and Convergent Learning in Stochastic Games, *IJCAI*, pp. 1021–1026 (2001).

[13] Klimov, O. and Schulman, J.: Roboschool, `https://blog.openai.com/roboschool/` (2017).

[14] Heinrich, J., Lanctot, M. and Silver, D.: Fictitious Self-Play in Extensive-Form Games, *ICML*, pp. 805–813 (2015).

[15] Heinrich, J. and Silver, D.: Deep Reinforcement Learning from Self-Play in Imperfect-Information Games, *arXiv:1603.01121* (2016).

[16] Shamma, J. S. and Arslan, G.: Dynamic fictitious play, dynamic gradient play, and distributed convergence to Nash equilibria, *IEEE Transactions on Automatic Control*, Vol. 50, No. 3, pp. 312–327 (2005).

[17] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O.: Proximal Policy Optimization Algorithms, *arXiv:1707.06347* (2017).

[18] Schulman, J., Levine, S., Abbeel, P., Jordan, M. and Moritz, P.: Trust region policy optimization, *ICML*, pp. 1889–1897 (2015).

[19] Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S. and Wu, Y.: OpenAI Baselines, `https://github.com/openai/baselines` (2017).

[20] Johanson, M., Waugh, K., Bowling, M. and Zinkevich, M.: Accelerating Best Response Calculation in Large Extensive Games, *IJCAI*, pp. 258–265 (2011).

[21] Lisy, V. and Bowling, M.: Equilibrium Approximation Quality of Current No-Limit Poker Bots, *AAAI Workshop on Computer Poker and Imperfect Information Games* (2017).

[22] Hausknecht, M. and Stone, P.: Deep Recurrent Q-Learning for Partially Observable MDPs, *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents* (2015).