モンテカルロ木探索を用いたハゲタカのえじきエージェントの評価

志村 伊生六 [1, a)]　　山口 文彦 [2, b)]

概要：モンテカルロ木探索は、シミュレーションを用いることで、ゲームにおける最善手を効率良く求めることができ、逐次手番ゲームにおける有効なアルゴリズムとして普及している。これを、行動を同時に選択する同時手番ゲームに用いる場合は、自分の行動を決定する際に他プレイヤの行動を観察することが出来ないため、逐次手番ゲームとは異なり、プレイヤ毎に求めた評価値をまとめた上でノードを選ぶ。本研究では、ハゲタカのえじきという同時手番ゲームの 2 人プレイで、モンテカルロ木探索の複数のバリエーションを用いた対戦エージェントを制作し、対戦における強さはどの程度か、理想的な戦略と考えられるナッシュ均衡戦略にどのくらい近づけるかを調べる。またハゲタカのえじきの 3 人プレイにおいて、2 人プレイのエージェントに用いたアルゴリズムを基にモンテカルロ木探索の手法を提案し、対戦における強さを調べる。

キーワード：モンテカルロ木探索，ハゲタカのえじき，同時手番ゲーム

1. はじめに

対象のゲームで最善手が何かを調べるとき、理論的にはゲーム木を全て探索することで、最善手が得られる。しかしゲーム木のサイズが大きくなると、全て探索することは極めて難しい。モンテカルロ木探索（MCTS）は、木探索においてシミュレーションを用いることにより、良い手を効率よく探索することができる手法である。探索空間が大きい囲碁などにおいて、著しい性能を発揮している[1]。しかし、MCTS においてゲーム木を用いた探索を行うとき、ゲーム木内の各ノードには評価値を一意的に付与して最善手を求める場合が多い。この方法は、一つの状態から何かしら行動をとって次の状態へ遷移するとき、その行動をとるプレイヤが一人ずつであるような、逐次手番ゲームにおいては素直に適用することが可能である。しかし、参加しているプレイヤの内複数人（全員も含む）が同時に行動をとるような、いわゆる同時手番ゲームへ適用する場合は、一つのノードに対して評価値が一意的なままだと、ノードを辿ってゲーム木を探索する際に自分以外の行動を不当に考慮してしまう。

本研究では、同時手番ゲームに対する MCTS の利用可能性について、ハゲタカのえじきという同時手番ゲームを題材に、MCTS を用いたエージェントを制作して、対戦における強さがどの程度かを勝率によって比較する。また用いた手法から求められた戦略が、どれほどナッシュ均衡戦略に近づけるかを、可搾取量と呼ばれる定量的

1 長崎県立大学
2 長崎県立大学
a) mc221001@sun.ac.jp
b) yamagu@sun.ac.jp

な指標によって比較する。

2. 関連研究

MCTS の同時手番ゲームへの適用例を紹介する。

Finnsson らは、General Game Playing（GGP）にて記述される単純な同時手番ゲームに対して MCTS を適用し、2007 年と 2009 年の GGP 大会（ The International General Game Playing Competition）にてトップの成績を収めた[2]。

Den Teuling らの研究では、同時手番ゲームである Tron を対象に、同時手番ゲームをあたかも逐次手番ゲームかのように扱って探索する MCTS について、通常の MCTS よりも MCTS-Solver が、優れた勝率を収めた[3]。変わって Lanctot らは、同じく Tron を対象に、MCTS のバリエーションとして DUCT を改良した Decoupled UCB1-Tuned（DUCB1T）を提案し、他に適用した MCTS のバリエーションの中でも、勝率の面で DUCB1T が最も優れた成績を収めたことを示した[4]。

また Lanctot らは、次にハゲタカのえじきの元となっているゲーム Goofspiel を対象に、MCTS のバリエーションとして OOS を提案し、3 章にて示した複数の MCTS のバリエーションの中で、勝率の面で RM と OOS が最も優れた成績を収めたことと、RM と OOS がナッシュ均衡に収束していくことを実験的に示した[5]。

3. 同時手番ゲーム

同時手番ゲームのモデルは、次の要素にて構成される[5]。

- 参加プレイヤの集合 N
  それぞれのプレイヤは$i \in N$と表記され、またNにはプレイヤの行動以外の偶然によって状態が遷移する偶然手番cも含んでいる。
- 状態の集合 S
  プレイヤが行動を決める状態D、偶然手番が発生する状態C、ゲームの終端状態Zで構成される。
- プレイヤがとる行動の集合 A
  プレイヤiの行動集合をA_iとするとき、同時手番ゲームでは、Aは各プレイヤの行動の直積となる。（例えばプレイヤ数が 2 である場合は、
  $$A = A_1 \times A_2$$
  となる。）
- 状態遷移関数 T
  現在の状態から全プレイヤの行動（各プレイヤの行動の組）が与えられたとき、遷移する次の状態を定義する。
- 偶然手番における状態遷移関数 Δc
  偶然手番によって遷移する次の状態を定義する。
- 効用関数 u_i
  プレイヤiの効用を定義する。定和ゲームにおいては
  $$\forall z \in Z, u_i(z) = k - u_{-i}(z)$$
  が成り立つ。
- ゲームの初期状態 s_0
  ゲーム開始時点の状態を表す。

例えば、典型的な同時手番ゲームとしてじゃんけんがある。また、じゃんけんを連続で行い、最終的な勝敗を決めるようなゲーム（グリコじゃんけんなど）も、状態の遷移はあるが、プレイヤの行動が同時に選択されるため、同時手番ゲームに分類される。

全てのプレイヤに対して、どのプレイヤもそれ以上効用を向上させることが出来ない状態をナッシュ均衡と呼ぶ。同時手番ゲームのように、プレイヤにとって見える情報が不確定であるような

ゲームの場合、ナッシュ均衡に至る行動を選択する戦略が、あらゆる戦略に対して最も負けないという意味で最善である。

プレイヤiの行動戦略はσ_iと表記され、戦略プロファイル$\sigma = (\sigma_i,\ \sigma_{-i})$が与えられたとき($\sigma_{-i}$はプレイヤ$i$以外の戦略を表す)、$\sigma$の下で終端状態$z$に到達する確率$\pi^\sigma(z)$は、偶然手番も考慮すると

$$\pi^\sigma(z) = \pi_i(z)\pi_{-i}(z)\pi_c(z)$$

と表せる。このとき、次の式を最適化する行動戦略の組は、ナッシュ均衡を保っている状態であるといえる。

$$V^* = \max_{\sigma_i \in \Sigma_i} \min_{\sigma_{-i} \in \Sigma_{-i}} \sum_{z \in Z} \pi^\sigma(z) u_1(z)$$

4. 同時手番ゲームのためのモンテカルロ木探索(SM-MCTS)

MCTS では、大きいゲーム木での探索においても最善手が得られるよう、以下の 4 ステップによるシミュレーションを繰り返し行って探索する。

① ノードの選択(Selection)
ノードに付与された評価値を元に優先度の高い子ノードを選択し、ゲーム木を降下する。

② ノードの展開(Expantion)
選択回数が閾値を超えたノードを展開し、ゲーム木を成長させる。

③ ノードの評価(Evaluation)
閾値を超えていない葉に到達したら、そこから終端状態までランダムシミュレーションを行う。

④ ノードの更新(Backpropagation)
①にて辿ってきたノードを引き返しながら、③によって得られた結果を反映する。

逐次手番ゲームでは、ある状態において行動をとるプレイヤは一人であるため、ノード一つに対して一つの評価値を付与すればよい。これを同時手番ゲームに用いる場合は、ゲーム木を探索する上でも、自分以外のプレイヤの行動を実際のゲームの流れと同じタイミングで考慮できるように、一つのノードに対して参加している各プレイヤ別の評価値を付与する。またプレイヤ一人の行動でノードを構築するのではなく、参加プレイヤそれぞれの合法手の組でノードを構築する。このようなゲーム木を用いて、用いられるモンテカルロ木探索のバリエーションを次に示す。

4.1. SM-UCT

SM-UCT は、モンテカルロ木探索の一手法として有名な UCT を、同時手番ゲームにおけるゲーム木に対して素直に実装した手法である。各ノードでは、プレイヤ別にそのノードにおける報酬の累積X_s^iと、ノードの選択回数の累積n_s^iを保持しておく。Selection では、まず保持した二つの値から子ノード毎に UCB 値を導出する。

$$UCB = \bar{X}_s^i + c\sqrt{\frac{\log n}{n_s}}, \text{where } \bar{X}_s^i = \frac{X_s^i}{n_s^i}$$

通常の UCT では、一番 UCB 値が高いノードを選ぶが、SM-UCT では、プレイヤの行動が該当する子ノード毎に UCB 値の和を取り、その値が高い行動の組に該当する子ノードを選ぶ。

Backpropagation では、Evaluation で得た効用u_iを元に、選んだノードに対して報酬の累積と選択回数の累積を更新する。

$$X_s^i = X_s^i + u_i, \quad n_s^i = n_s^i + 1$$

繰り返しシミュレーションを行ったのち、最終的にルート(根)ノードにおいて、子ノードの選択回数を Selection と同じ要領で合法手毎に和を取り、その値が一番大きい手を一意的に選ぶ。これは、最終的に自分にとって最も報酬を最大化することを意味している。

4.2. Decoupled UCT

Decoupled UCT(DUCT)は、SM-UCT と同じく UCT をベースとした手法である[5]。SM-UCT が、子ノード毎に値を保持していたのに対して、Decoupled UCT(DUCT)は、各ノードにおいて、プレイヤの合法手毎に報酬の累積と、ノードの選択回数の累積を保持しておく。ノードを選んで探索する際は、プレイヤ別に保持した二つの値から UCB 値を導出し、プレイヤiにとって UCB 値が最も高い手a_iを求める。

$$a_i = \underset{a \in A_i(s)}{\operatorname{argmax}} \left\{ \bar{X}^i_{s,a} + c\sqrt{\frac{\log n_s}{n_{s,a}}} \right\},$$

$$\text{where } \bar{X}^i_{s,a} = \frac{X^i_{s,a}}{n^i_{s,a}}$$

これをプレイヤ別に行い、それぞれの行動aの組に該当するノードを選ぶ。

Backpropagation では、選んだ行動の履歴を遡り、式()に準じて$X^i_{s,a}$と$n^i_{s,a}$を更新する。最終的にルート(根)ノードにおける合法手のうち選ばれた回数が一番多い手を一意的に選ぶ。また、合法手それぞれの手が選ばれた回数の割合に基づいて、混合戦略的に着手する方法もある。前者を DUCT(max)、後者を DUCT(mix)と呼ぶ。

4.3. Exp3

Exp3 では、DUCT と同じように、各ノードにおいて報酬の累積と選択回数の累積を保持しておく。ノードを選ぶ際は、各プレイヤが合法手のそれぞれの手に関して、次の式によって混合戦略$\sigma^t_i(s, a_i)$を導出し、それに基づいてノードを選ぶ。

$$\sigma^t_i(s, a_i) = \frac{(1-\gamma)\exp\left(\eta\omega^i_{s,a_i}\right)}{\sum_{a_i \in A_i(s)} \exp\left(\eta\omega^i_{s,a_i}\right)} + \frac{\gamma}{|A_i(s)|}$$

$$\eta = \frac{\gamma}{|A_i(s)|}, \text{ and } \omega^i_{s,a_i} = \hat{x}^i_{s,a} - \max_{a' \in A_i(s)} \hat{x}^i_{s,a'}$$

Backpropagation では、n^i_sの更新に加え、$\hat{x}^i_{s,a}$をシミュレーションによって得られた効用を用いて、次の式に従って更新する。

$$\hat{x}^i_{s,a} = \hat{x}^i_{s,a} + \frac{u_i}{\sigma^t_i(s, a_i)}$$

これは、選ばれた手に関して、一度選ばれただけでなく、すべての反復の報酬の合計を推定している。繰り返しシミュレーションが終わったら、最終的にルートノードにおける合法手のうち各手の累積選択回数の割合で、混合戦略を構築し、それに従って着手する。

4.4. Regret Matching

Regret Matching(RM)は、元々ナッシュ均衡を求めるためのアルゴリズムとして、ポーカーなどの状態数が大きいゲームにおいて効果を発揮していたアルゴリズムであり、それを MCTS のシミュレーションにおける各反復に応用する[5]。これは、とった行動に対して「あの行動をとっておけばよかった」という regret(後悔)を出来るだけ小さくしていくように、手を選んでいく。各ノードにおいては、プレイヤ別に累積報酬と累積選択回数に加

え、それぞれのプレイヤにとっての合法手に対する regret の累積と、そのノードから合法手を選ぶ際に用いられた混合戦略について、今までとられた戦略の平均を保持する。

その探索に用いる混合戦略は、おおむね各合法手の正の regret の割合とする。ただし、累積 regret が 0 である場合は、一様ランダムにノードを選択する。

$$\sigma_i^t(s,a_i) = \frac{r_s^i[a]}{R_{sum}^+}$$

$$if\ R_{sum}^+ > 0 \text{ otherwise.} \frac{1}{|A_i(s)|}$$

$$\text{where } R_{sum}^+ = \sum_{a \in A_i(s)} r_s^{i,+}[a]$$

これにより、累積 regret が高い手を優先的に選ぶようになる。式中のγは、探索を確実にするためのハイパーパラメータである。更新の際は、シミュレーションによって得られた利得と、各ノードにおける期待報酬との差を regret として累積 regret を更新するほか、各ノードにおける戦略の平均も更新する。

$$\forall a'_i \in A_i(s),$$
$$r_s^i[a'_i] + (reward(a'_i, a_{-i}) - u_i)$$
$$\bar{\sigma}_s^i[a] = \bar{\sigma}_s^i[a] + \sigma_s^i[a]$$

繰り返しシミュレーションを行った後は、ルートノードの合法手において、それぞれの手の平均戦略の割合で構築された混合戦略に従って着手する。

4.5. Online Outcome Sampling

Online Outcome Sampling（OOS）は、二人零和不完全情報ゲームなどにおいて、サンプリングを用いることでナッシュ均衡を求める Monte Carlo CFR（MCCFR）と、MCTS を組み合わせた手法である[5]。MCCFR は、あらかじめ対象のゲームにおける最初の状態をルートノードとしたゲーム木を探索するが、OOS は MCTS と同様に、ゲームの進行を観測し、プレイしている現在の状態をルートノードとした部分ゲーム木を探索する。

この手法で行われるシミュレーションでは、ノードの選択や値の更新について、探索対象となるプレイヤi_{exp}がシミュレーション間で交互に入れ替わる。ノードの選択段階では、まず RM と同じ方法（式 ）にて混合戦略$\sigma_i^t(s,a_i)$を構築し、ノードを選択するときに従う混合戦略は、探索プレイヤ$i = i_{exp}$と相手のプレイヤjで次のように決める。

$$p_{s,a_i} = \frac{\gamma}{|A_i(s)|} + (1-\gamma)\sigma_i^t(s,a_i)$$
$$p_{s,a_j} = \sigma_j^t(s,a_j)$$

ノードの更新段階では、$\sigma_i^t(s,a_i)$と$\sigma_j^t(s,a_j)$が与えられたときの合法手毎の期待値$v_s^i[a]$を計算する。また戦略プロファイルσが与えられたときの期待値$v_{s,\sigma}^i$を計算する。

$$v_s^i[a] = \sum_{a' \in A_j(s)} \sigma_j^t(s,a') X_{s,a'}^j$$

where $X_{s,a}^j$

$$= \begin{cases} u_i & \text{if } \{a,a'\} \text{ were selected} \\ \frac{X_{s'}}{n_{s'}} & \text{otherwise., where } s' = T(s,a,a') \end{cases}$$

$$v_{s,\sigma}^i = \sum_{a \in A_i(s)} \sigma_i^t(s,a)\, v_s^i[a]$$

Backpropagation では、$v_s^i[a]$と$v_{s,\sigma}^i$を用いて累計 regret $r_s^i[a_i]$と戦略平均$\bar{\sigma}_s^j[a_j]$を更新する。

$$r_s^i[a_i] = r_s^i[a_i] + \frac{1}{\pi}\left(v_s^i[a_i] - v_{s,\sigma}^i\right)$$

$$\bar{\sigma}_s^j[a_j] = \bar{\sigma}_s^j[a_j] + \frac{1}{\pi}\sigma_j^t(s, a_j)$$

πは探索方針を決めるパラメータであり、ノードを選択して一段ずつ降下していく際に、$\pi = \pi p_{s,a_i}$として更新される。

繰り返しシミュレーションを行った後は、ルートノードの合法手において、それぞれの手の平均戦略の割合で構築された混合戦略に従って着手する。

5. ハゲタカのえじき

ハゲタカのえじきは、Alex Randolph 氏が作った 2 人から最大 6 人でプレイ可能なカードゲームである[7]。本来のルールでは、プレイヤは 1 から 15 の数字が書かれた 15 枚の数字カードを手札として持ち、-5 から 10 の数字（0 は除く）が書かれた 15 枚のハゲタカカードを、伏せた状態で山札として置く。まずハゲタカカードを一枚めくり、そのあとプレイヤは同時に数字カードを一枚表にして場に出す。そして次のルールに従って、ハゲタカカードを獲得するプレイヤを決める。

① めくられたハゲタカカードの数字が正の数だった場合、場に出た数字カードのうち一番数字が大きい数字カードを出したプレイヤがハゲタカカードを獲得する。

② めくられたハゲタカカードの数字が負の数だった場合、場に出た数字カードのうち一番数字が小さい数字カードを出したプレイヤがハゲタカカードを獲得する。

③ 一番大きな（小さな）数字カードが 2 枚以上場に出されていた場合は、次に大きな（小さな）数字カードを場に出したプレイヤがハゲタカカードを獲得する。これは、2 番目以降の大きさの数字カードについても同様に起こりうる。

④ ③のルールが場に出した数字カード全てにおいて発生し、誰もハゲタカカードを獲得できなくなった場合は、めくられたハゲタカカードはそのまま残して、次にめくるハゲタカカードと併せて一緒に奪い合う。

以上の流れを 15 枚のカードがなくなるまで行い、獲得したハゲタカカードの数字の合計をそのままポイントとして、一番獲得ポイントが高いプレイヤが勝利する。

6. 実験概要

本研究では、ハゲタカのえじきの 2 人プレイにおいて、3 章で紹介した MCTS の複数のバリエーションを用いた対戦エージェントを制作し、そのエージェントの性能を、勝率と可搾取量という二つの指標で評価する。また 3 人プレイ用のエージェントを改良し、その勝率を調べる。

6.1. エージェントの評価方法

勝率は、制作したエージェントそれぞれについて、ランダムに行動とるランダムエージェント（RAND）と、めくられたハゲタカカードの数字の大きさとほぼ同じ強さのカードを出す特殊プレイヤ（SP）との対戦を行ったときの勝率、またそれぞれのエージェント同士の総当たり戦を行い、その勝率を比較する。

また、ゲームの最初の状態から繰り返しシミュレーションを行って戦略を更新し、用いた手法から求められる戦略プロファイルに対する可搾取量がどう変化するかを調べる。可搾取量とは、戦略プロファイルがナッシュ均衡にどれほど近いかを示す値である。零和ゲームにおける可搾取量$\epsilon(\sigma)$は、次の式で定義される[9]。

$$\epsilon(\sigma) = \sum_{i \in N} b_i(\sigma_{-i})$$

$$\text{where} \quad b_i(\sigma_{-i}) = \underset{\sigma_i}{\operatorname{argmax}}\, u_i(\sigma_i, \sigma_{-i})$$

戦略プロファイルがナッシュ均衡状態であるとき、可搾取量は 0 になる。つまり、この値が小さければ小さいほど、ナッシュ均衡に近いといえる。今回は、制作したエージェントのうち DUCT（mix）、Exp3、RM、OOS の 4 つで、それぞれ戦略更新を行い、可搾取量の変化を調べる。

6.2. 3 人プレイにおける SM-MCTS の実装

まず 3 人以上のプレイヤが参加する多人数プレイのための MCTS として、MCTS-MAX^n という手法がある[8]。これは、参加しているプレイヤがそれぞれ評価値を保持しており、行動を選ぶ際は自身の評価値が一番高い行動を選ぶというものである。この手法において、一つのノードに各プレイヤ別の評価値が記録されていることに着目し、3 章で紹介した SM-MCTS のバリエーションでも、保持する評価値をそのままプレイ人数分に拡張して素直に実装出来ると考えた。この拡張手法についてもそれぞれエージェントを制作し、2 つの RAND との 3 人対戦を行い、勝率を比較する。

7. 実験結果

対戦では、プレイ人数に関わらず、各エージェントにおいて一つ手を決めるときに繰り返し行われるシミュレーションを 100000 回行い、計 1000 試合行った。

戦略更新によって可搾取量を求める際は、繰り返し行われるシミュレーションを 100000 回行った。可搾取量を求める対象の戦略プロファイルについては、ここでは、一つの状態毎にシミュレーションをやり直して戦略を求めたものではなく、ゲームの最初の状態をルートノードとして、シミュレーションによって求められた戦略プロファイルに従うものとする。

全ての手法において、ノードの選択や更新で用いられる効用には、終端状態での勝敗（勝ち:1 引き分け:0.5 負け:0）を採用した。

RAND

手法	勝率
Basic-UCT	91.5%
DUCT(max)	76.5%
DUCT(mix)	76.0%
Exp3	73.4%
RM	83.1%
OOS	82.1%

SP

手法	勝率
Basic-UCT	77.1%
DUCT(max)	64.1%
DUCT(mix)	57.5%
Exp3	45.5%
RM	63.3%
OOS	58.5%

表 1：RAND、SP との対戦結果

7.1. 2 人プレイにおける対戦勝率

RAND と SP をそれぞれ相手にした対戦勝率を表 1 に、総当たり戦における示す。引き分けは 0.5 勝として計算している。

まず全てのエージェントが、RAND、SP それぞれに対して 50%以上の勝率で勝ち越した。その中でも SM-UCT が、RAND と対して 91.5%、SP に対して 77.1%と最も良い成績を収めた。また SP に対する勝率の面では、Exp3 が全てのエージェ

	Basic-UCT	DUCT (max)	DUCT (mix)	Exp3	RM	OOS
Basic-UCT	／	23.5%	36.5%	58.5%	31.5%	33.3%
DUCT(max)	77.5%	／	54.0%	64.5%	47.8%	44.7%
DUCT(mix)	63.5%	46.0%	／	62.0%	44.5%	48.9%
Exp3	41.5%	36.5%	38.0%	／	26.0%	30.2%
RM	68.5%	53.2%	55.5%	74.0%	／	50.5%
OOS	66.7%	55.3%	51.1%	69.8%	49.5%	／

表 2:総当たり戦での勝率

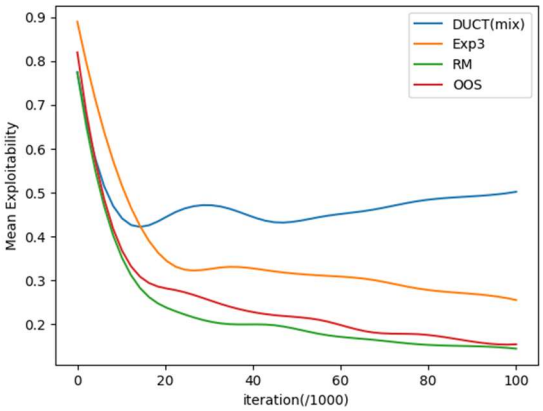

図 1:戦略更新による可搾取量の変化

手法	勝率
DUCT(max)	66.3%
DUCT(mix)	55.9%
Exp3	53.1%
RM	70.1%

表 3:2 つの RAND との対戦結果

ントの中で最も悪い成績となった。

エージェント同士の総当たり戦では、RM が勝率の面で一番勝ち越しており、次に OOS が勝ち越しているという結果になった。ただ RM と OOS との対戦における勝率は、数値的には RM の勝率が 50.5%と勝ち越しているものの、エージェントの強さとしてはほぼ互角であることが確認できた。このことから、全てのエージェントの中で RM と OOS が最も強かったという結果になった。これは、[5]の研究における Goofspiel での実験結果とほぼ同様の結果となった。

7.2. 2 人プレイにおける可搾取量の変化

可搾取量を計算するためには、ゲームにおける全ての状態について、到達確率と期待値を計算しなければならないため、本来の 15 枚ルールにおける可搾取量、現実的な時間で計算することは難しい。今回は縮小版である 5 枚バージョンで可搾取量を計算した。その結果を図 1 に示す。

全てのエージェントは、戦略更新を 100000 回行ったうち、最初の方では可搾取量を減少させることに成功しているものの、DUCT(mix)についてはその後可搾取量を
増えていることが確認できた。また他のエージェント Exp3、RM、OOS はそれぞれ戦略更新を重ねるにつれて可搾取量を減少させることが出来ており、特に RM と OOS は、Exp3 よりも可搾取量の減少の幅が大きかった。この結果と 7.1 節の勝率の結果と併せると、[5]の研究結果と同様に、ハゲタカのえじきの二人プレイにおいて、今回採用したエージェントの中では、RM と OOS が最も良い性能を発揮したことが分かった。

7.3. 3 人プレイ

続いて DUCT(max)、DUCT(mix)、Exp3、RM において、2 人プレイ用から 3 人プレイ用に拡張したエージェントを、2 つの RAND と対戦させた結果を表 3 に示す。

勝率の面で、全てのエージェントが、2 つの RAND に勝ち越した。中でも DUCT(max)と RM の勝率が優れており、特に RM の勝率は 70.1%と、RAND 相手であれば強いエージェントであることが分かった。対して DUCT(mix)と Exp3 は、どちらも勝率が 50%台であり、DUCT(max)と RM に比べて弱いことが分かった。

8. まとめ

本研究では、同時手番ゲームにおける MCTS の利用可能性を評価するために、カードゲームのハゲタカのえじきを対象に、MCTS の複数のバリエーションを用いたエージェントを制作し、対戦における勝率と、戦略更新での可搾取量の変化を調べた。まず 2 人プレイにおいては、RAND 相手、及び互いのエージェント同士での対戦勝率の面で、RM と OOS が最も良い成績を収めた。加えて戦略更新での可搾取量の減少幅についても、RM と OOS が最も大きかったことから、この二つのエージェント及び用いた手法が、ハゲタカのえじきに対して有効である可能性が示唆された。また 2 人プレイ用の手法を 3 人プレイ用に拡張し、それぞれ実装したエージェントを 2 つの RAND 相手に対戦させた結果、DUCT(mix)と RM が他のエージェントと比べて良い勝率を収めたことから、3 人プレイにおいても、RM が有効である可能性が示唆された。

今後の課題として、3 人プレイでの OOS の実装や、3 人プレイでの戦略更新における可搾取量の変化を調べることが挙げられる。特に 3 人プレイにおける可搾取量については、推論的には計算が可能なものの、厳密な定義がまだ出来ていないため、理論的な説明を行う必要がある。また実験においてシミュレーションにかかる時間が非常に長く(3 人プレイの RM では 12 時間程度)、人間相手との対戦プレイヤとしては、今回の手法が向かない問題があったため、その処理時間を削減する工夫が必要である。

参考文献

[1] Rémi Coulom: Efficient selectivity and backup operators in Monte-Carlo tree search, International Conference on Computers and Games, pp.72-83(2006)

[2] Finnsson, H.: Simulation-Based General Game Playing. Ph.D. thesis, Reykjavik University(2012)

[3] Niek G.P. Den Teuling, Mark H.M. Winands: Monte-Carlo Tree Search for the simultaneous move game Tron. Proceedings of Computer Games Workshop (ECAI), pp.126-141(2012)

[4] Mark Lanctot, Christopher Wittlinger, Mark H.M. Winands, Niek G.P. Den Teuling: Monte Carlo Tree Search for Simultaneous Move Games: A Case Study in the Game of Tron. Proceedings of the 25th Benelux Conference on Artificial Intelligence, BNAIC 2013(2013)

[5] Marc Lanctot, Viliam Lisy, Mark H.M. Winands: Monte Carlo tree search in simultaneous move games with applications to Goofspiel. Workshop on Computer Games, pp.28-43(2013)

[6] 大町洋, 池田心: 同時進行ゲームのためのモンテカルロ木探索, ゲームプログラミングワークショップ 2012 論文集, pp.197-

204(2012)
[7] ハゲタカのえじき - メビウスゲームズ, https://www.mobius-games.co.jp/mobiusgames/Hagetaka.html
[8] 大西 紘史, 保木 邦仁: 4 人でプレイする Blokus の AI プレイヤの強化学習, 研究報告ゲーム情報学, pp.1-8(2020)
[9] 河村 圭悟, 鈴木 潤, 鶴岡 慶雅: 未知環境における多人数不完全情報ゲームの戦略計算, ゲームプログラミングワークショップ 2017 論文集, pp.80-87(2017)