同時進行ゲームのためのモンテカルロ木探索

大町洋　　池田心
北陸先端科学技術大学院大学
{h.omachi, kokolo} @jaist.ac.jp

近年，チェスや囲碁，将棋など様々なボードゲームの AI が研究され，既に人間のプロのレベルに到達しているものも存在する．しかし，ボードゲームは着手の同時公開の困難からか交互ゲームが多く，ジャンケンなどのそれぞれのプレイヤが着手を決定してから同時に公開する同時進行ゲームの AI 開発に関する研究は少ない．本稿では，同時進行ゲームの AI 開発の足掛かりとして，ナッシュ均衡を考慮した確率的なノード探索をモンテカルロ木探索に適用する手法を提案する．提案手法は二人零和完全確定情報同時進行ゲームである同時進行 Triomineering にて，モンテカルロ法に対して優位な結果を示すことに成功した．

Monte-Carlo Tree Search for Synchronized Games

HIROSHI OMACHI,　　KOKOLO IKEDA
Japan Advanced Institute of Science and Technology

Recently, many AIs are researched and some of them can already compare favorably with human strong players. But most of their targets are Sequential Games because of difficulty of synchronized move on any board games. In this paper, we propose a method which is considering Nash-equilibrium Monte-Carlo Tree Search for applying to Synchronized Games like roulette. In experiments in Synchronized Triomineering which is perfect information zero-sum synchronized game, our method gets an advantage over Monte-Carlo algorithm.

1. はじめに

近年, 囲碁などの交互ゲームの AI 開発において, モンテカルロ木探索（MCTS）が幅広く利用されている．一方，ジャンケンやルーレット，ポケットモンスターなどのビデオゲームの様に, 互いのプレイヤの着手が決定してから同時に公開するような同時進行（Synchronized. 同時着手ともいう）ゲームに対する研究はあまり進んでいない．一般に，ボードゲームでは着手の同時公開の困難からか同時進行ゲームはあまり存在しないが, ビデオゲームには同時公開が容易であることから多くの同時進行ゲームが存在しており，強い AI を開発するためのアプローチが必要である．

小森らは同時進行化した Domineering にモンテカルロ法（MC 法）を適用した[1]が，単に深さ 1 までしか探索しないということのみならず, 相手の手も含めて全ての合法手に対して等確率にプレイアウトを行うため, 相手の着手によっては極端に自分が不利になるような着手にも着手してしまう問題がある．また，多くの同時進行ゲームにおいてプレイヤは混合戦略による確率的な着手が必要となるが, MC 法では混合戦略に基づく着手が困難となる問題もある．

そこで本研究では, 同時進行ゲームにおいてより効率的な探索を行うことを目標とし, ナッシュ均衡となる混合戦略を考慮した UCB 探索を提案し，同時進行 Triomineering を対象として提案手法の実装と評価を行った．通常の UCB 探索では UCB1 値が高いノードを決定的に選択するが, 同時進行ゲームでは混合戦略による確率的な着手が必要となるので, UCB1 値を利得としてナッシュ均衡となる混合戦略の近似を求め確率的な探索を行うことで，互いにとって有望な手の組み合わせを重点的に調べつつも他の手の見落としが無い探索を実現した．

実験の結果, 提案手法はモンテカルロ法に対しては優位な結果を得た．また，それぞれのプレイヤの UCB1 値が高い手同士を探索する単純 UCB 探索に対しては，優位な結果を得られなかったものの，非零和ゲームにおける囚人のジレンマ的なインスタンスにおいては，提案手法と単純 UCB 探索との間で探索結果の差異が現れることを示した．

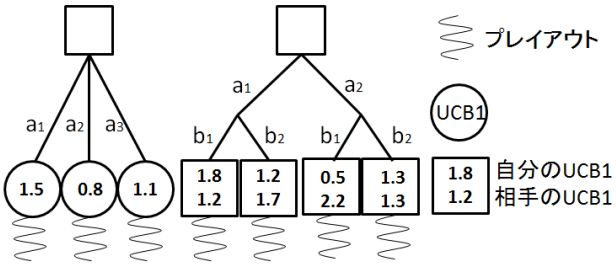

Figure 1 深さ 1UCB 探索の例

2. 同時進行ゲームにおける従来手法の問題

交互ゲームにおいて，通常の UCT は UCB1 値が高いノードを展開することで効率的に探索を行う．Figure 1 の（左）にその例を示す．この例の場合，UCB1 探索ではa_1が選択される．一方で，同時進行ゲームは自分と相手の着手の組み合わせにより局面が推移するため, 自分の合法手 1 つに対して複数のノードが生成され, 自分の着手選択により推移する局面が確定的ではない．Figure1（右）にその例を示す．この例の場合，相手がb_1 , b_2のどちらを探索するかによって, 自分が探索するべきノードが異なる．

MC 法はランダムに着手を決めることから, 相手の初手に良い手があってもそこに探索が集中しない．また，通常の MC 法や UCT は自分の標本平均利得や探索回数のみから決定的に手を選択するため, 同時進行ゲームに必要な混合戦略を取るのが難しい．この問題を解決するためには，自分の合法手と相手の合法手を組としてノードを作成し探索する必要があるが, その際にはどの様に探索するノードを決定するかが難しい問題となる．

3. 提案手法

同時進行ゲームにおける探索ノード決定の問題を解決するために, 本研究ではナッシュ均衡となる混合戦略を近似により求め, その混合戦略に基づき確率的にノードを探索する手法を提案する. アルゴリズム全体の概略を Figure 2 に示す．このアルゴリズムは, 自分の合法手と相手の合法手の組をエッジとしてノードを生成し, 各ノードについて自分と相手の UCB1 値からナッシュ均衡となる様な混合戦略を近似により求めることで, 互いにとって有望な手の組み合わせを重点的に探索する．本来，UCT の深さは 1 である必要はないが，まず MC 法との比較を行うために深さ 1 に限定して実装した. このアルゴリズムを便宜上 Mixed Strategy applied UCB（MSUCB1）とする．さらに深さに関する拡張を行った MSUCB1.5 に関しては，後述する．

1： 自分の合法手と相手の合法手の組をエッジとし，深さ 1 の探索木を生成する

2： 生成された全てのノードについて, 1 度ずつプレイアウトを行う

3： 互いの合法手を戦略，**UCB1 値**を利得とし，ナッシュ均衡となる混合戦略を近似する

4： 混合戦略に基づき, 探索するノードを確率的に決定する

5： 選択したノードでプレイアウトを行う（深さ 2 以降を同様に分岐させても良い）

6： 3～5 を一定回数繰り返す

7： 探索木の深さ 1 の各ノードについて，互いの合法手を戦略，**標本平均利得**を利得とし，ナッシュ均衡となる混合戦略を近似する

8： 混合戦略に基づき，着手を選択する

Figure 2 提案手法のアルゴリズム

3.1 ナッシュ均衡の近似

提案手法ではナッシュ均衡となる混合戦略を, Figure 3 のアルゴリズムに従い計算している．このアルゴリズムによってナッシュ均衡となる混合戦略が正しく近似されることは, 予備実験にて確認している．

1： 自分と相手それぞれの利得表を入力

2： 各合法手の選択確率を等確率に初期設定

3： 自分の合法手の利得係数を計算

4： 利得係数が高い合法手の選択確率を増加, 低い合法手の選択確率を減少させる

5： 自分の合法手の平均選択確率を更新

6： 3～5 と同様に相手の合法手の選択確率を変更し，平均選択確率を更新

7： 3～6 を一定回数繰り返す
途中で平均選択確率が変更しなくなった場合は収束したとみなして終了

8： それぞれの平均選択確率を出力

Figure 3 ナッシュ均衡近似アルゴリズム

このアルゴリズムにおける利得係数とは, プレイヤ A のある合法手に関して，プレイヤ B の着手確率を考慮した際の期待利得への係数である. プレイヤを A,B とし，p（i）＝プレイヤ A の合法手iの着手確率（$i=1\sim m$）, q（j）＝プレイヤ B の合法手jの着手確率（$j=1\sim n$），u（i,j）＝プレイヤ A の合法手i, プレイヤ B の合法手jの組み合わせによるプレイヤ A の利得とすると，プレイヤ A の各着手における利得係数coe（i）は

$$\mathrm{coe}(i)=\sum_{j=1}^{n}\mathrm{q}(j)\times\ \mathrm{u}\ (i,j)$$

となる．また，プレイヤ A の総期待利得 gain は,

$$\mathrm{gain}=\sum_{i=1}^{m}\mathrm{p}(i)\times\mathrm{coe}(i)$$

となる．ここで, 利得係数が高いということはその合法手を着手した際に得られる利得が大きいということである. そのため, 利得係数が最大の合法手の選択確率を上昇させることは, 自身の期待利得を上昇させることに繋がる. この操作を自分と相手のそれぞれの着手確率に関して交互に行うことで, アルゴリズム中の平均着手確率をナッシュ均衡となる混合戦略値に近似させることが可能である.

4. 同時進行 Triomineering

同時進行 Triomineering とは，Blanco らが提案した交互ゲームである Triomineering[4]を同時進行化したもの[2]である．Triomineering とは，プレイヤが Vertical（V）と Horizontal（H）に分かれ，格子状のボードに 3×1 のブロックを交互に着手し，着手不能となったプレイヤの敗北となるゲームである．着手の際には V は垂直に，H は水平にしか着手出来ない. また既に埋まっているセルに被る様な着手は出来ないが，同時進行 Triomineering では同時進行化により発生する 1×1 セルの重複は許容されている． 5×5 ボードの例を Figure 4 に示す．灰色のセルはすでにブロックが埋まったセルを表し，黒いセルは 1×1 の重複が発生したセルである．この例では互いに次の着手が不能であるため, 引分けとなっている.

また本研究において重要なのは, 行数 m, 列数 n, m + n≦14 のボードに対して理論値が求められている[3]ことである．そのため，実験の結果を理論値と比較することで，性能を評価することが出来る.

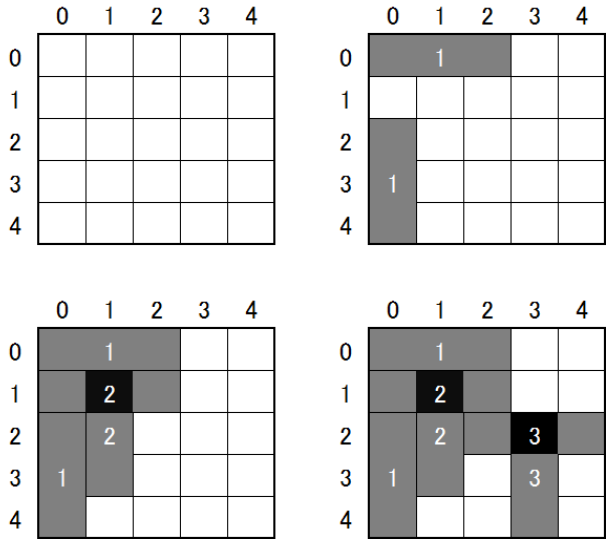

Figure 4 同時着手 Triomineering

また, このゲームはランダムプレイアウトで局面が収束するため，Monte-Carlo 的なアプローチが有効であることも重要である.

5. 関連研究

提案手法の実験の前に, 関連研究として小森らによる同時進行ゲームにおける MC 法の実験[1]を紹介する．この研究では，同時進行 Domineering というゲームを対象に, MC 法の実装及び評価を行っている．同時進行 Domineering とは本稿で紹介した同時進行 Triomineering と同様のルールのゲームであるが，使用するブロックが 2×1 であるという点で異なっている.

小森らの研究では, 同時進行 Domineering に単純な MC 法の AI を実装した．この AI は，探索は等確率なランダム探索を行い, 最終的な着手の決定は平均利得が最大の手としている.

また, 自己対戦の結果と互いのプレイヤが最善の手を指した場合の理論値とを比較し, MC 法の有効性を示しているが，同時進行ゲームに単純な MC 法を利用するには，2 章で述べた様に相手にとって良い手を考慮せずに探索及び着手の決定を行うという問題点がある.

6. 実験

6.1 AI の実装

本研究では，4 章で紹介した同時進行 Triomineering のシミュレータを作成し, その中に各

Table 1 実装した AI

名前	手法
Monte-Carlo（MC）	モンテカルロ探索
Simple UCB 0.5	自分の手の UCB 値のみを考慮する UCB 探索
Simple UCB 1	自分と相手の手のそれぞれの UCB 値を考慮する UCB 探索
Mixed Strategy UCB 1（MSUCB1）	深さ 1 まで探索する混合戦略を考慮した UCB 探索
Mixed Strategy UCB 1.5（MSUCB1.5）	深さ 1.5 まで探索する混合戦略を考慮した UCB 探索

手法の AI を実装して対戦を行なった．実装した AI は Table 1 に示す 5 種類である．

実装した AI は全てプレイアウトを必要とするが，本研究では，合法手の中から”相手の着手を妨害しない手”を削除するヒューリスティックを作成し，全ての AI がそのヒューリスティックを用いたランダムプレイアウトを行っている．また，正方形のボードの初手に限り，対称性を考慮して自分の合法手を削減するヒューリスティックも実装している．

また，最終的な着手の決定はそれぞれ，MC は平均利得最大の手を，SimpleUCB は探索回数最大の手を，MSUCB は平均利得からナッシュ均衡となる混合戦略的を求め，それに基づき着手を選択する．

6.2 提案手法と MC との比較

まず，2 章と 5 章で述べた MC の問題点が改善されているかを確認するため，6×6 ボードでの対戦実験と，提案手法が 4×4 ボードでどの様な探索を行っているかを調査した．

対戦実験におけるパラメータは，MC の Playout 数＝10,000/100,000，MSUCB1 の Playout 数＝10,000，UCB1 値のパラメータ c＝0.5 とし，MC の Playout 数ごとに 1,000 試合ずつ対戦させた．なお，プレイヤ V を MSUCB1，プレイヤ H を MC としている．6×6 ボードにおいて，プレイヤ V と H の間に有利不利は存在せず，互いが最適な着手を選び続けた場合，試合結果は必ず引分けになる．

実験結果を Table 2 に示す．1 ゲームごとの結果は V の勝利－引分け―H の勝利のいずれかであり，表の中の数字はそれぞれの結果が起こった回数を表す．Table 2 を見ると，提案手法である MSUCB1 は明らかに MC に対して優位な結果を得ていることがわかる．また，MC は Playout 数を 10 倍にしても性能が向上している様子が見られず，既に探索回数による性能上昇の上限に達していると考えられる．それなのにもかかわらず MSUCB1 が MC に対し優位な結果を示しているのは，探索時に相手の良い手を考慮していることと，最終的な着手の決定方法の違いからであると考えられる．

Table 2 6×6 ボードでの MSUCB1 vs MC

V＼H	MC	
MSUCB1	10,000	100,000
10,000	441-486-73	455-469-76

V-D-H

Table3 4x4 ボードの初手の MSUCB1 探索

V＼H	(0, 0)	(0, 1)	(1, 0)	(1, 2)	(2, 0)	(2, 1)	(3, 0)	(3, 1)	sum
(0, 0)	7	7	29	60	44	60	6	6	219
(0, 1)	31	34	2417	2417	2417	2417	24	24	9781
sum	38	41	2446	2477	2461	2477	30	30	10000

次に，提案手法である MSUCB1 の探索の調査を行う．4×4 ボードにおいてプレイヤ V を MSUCB1 とし，Playout 数＝10,000 で初手の探索回数を示したのが Table 3 である．ボードは 0～3 の 4 行 4 列で構成されており，合法手はそれぞれ 8 つずつある．ただし，V に関しては前述したヒューリスティックのよりノードを削減しており，2×8 のテーブルになっている．説明は省略するが，このゲームにおける初手の正着はそれぞれ，V=（0,1），H=（1,0）/（1,2）/（2,0）/（2,1）である．結果を見ると，”それぞれのプレイヤにとって正着である手の組”に探索が集中していることと，有望ではない”手の組”にも低い確率で探索が行われていることが確認できる．このことから，提案手法である MSUCB1 は，効率的かつ探索漏れがない様な探索を実現できていることが確認できる．

これらの結果から，提案手法である MSUCB1 は，MC の問題点であった探索の効率性と，最終的な着手選択の 2 つに対して，有効であると考えられる．

6.3 探索深さによる影響の調査

2 つ目の実験として，深さ 1 に限定して作成した MSUCB1 について，深さ 2 では自分の UCB1 値のみを考慮した探索を行う MSUCB1.5 を作成した．MSUCB1.5 の例を Figure 5 に示す．この AI を MSUCB1 と対戦させることで，提案手法の性能に探索深さが与える影響を調査した．対戦は 6×6 のボードにて，プレイヤ V を MSUCB1.5，プレイヤ H を MSUCB1 とし，それぞれ Playout 数＝10,000，

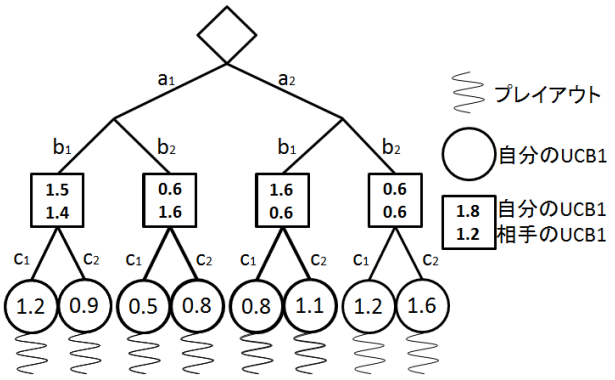

Figure 5 MSUCB1.5 探索の例

UCB1 値のパラメータ c＝0.5 で統一して 1,000 試合対戦させた．実験の結果は，V-D-H＝142-805-53 であり，MSUCB1.5 が有意に勝ち越していることが分かる．このことから，提案手法である MSUCB 探索は, 探索深さを拡張させることで更なる性能向上の可能性が見いだせる．

6.4 UCB パラメータによる影響の調査

3 つ目の実験として, 提案手法である MSUCB 探索において, UCB1 値計算の際のパラメータ c を変更させることにより，どのように AI の性能が変化するかを調査した．本研究では，以下の式を用いて単純な UCB1 値を計算している．なお，$\overline{X_{i,j}}$は合法手の組(i,j)の標本平均利得, nは総探索回数, $n_{i,j}$は合法手の組(i,j)を探索した回数を表している．

$$\mathrm{ucb1}(i,j) = \overline{X_{i,j}} + c \times \sqrt{\frac{2 \times \log n}{n_{i,j}}}$$

上記の式から分かる通り，c の値を大きくするほど未探索のノードを探索しやすくなり, 逆に小さくするほど標本平均利得が高いノードを選択しやすくなる．そこで, パラメータ c を変化させることで探索中のノード選択の優先度を変化させ，AI の性能にどのような影響が出るか実験により調査する．

実験は, 6×6 ボードにてプレイヤ V を MSUCB1, プレイヤ H を MC とし，それぞれの Playout 数＝10,000 として，いくつかの MSUCB1 の c の設定値ごとに 1,000 試合ずつ対戦させた. 実験結果を Table 4 に示す.

結果を見ると，c＝0.3～0.7 の結果及び利得の差はそれぞれ誤差の範囲内であり, 有意な差は見られなかった．しかし，それ以外のパラメータでは性能が有意に悪くなっており，UCB1 パラメータ c は大

Table 4 UCB1 パラメータ c による影響

c	V-D-H	gain
0	227-493-280	0.474
0.1	268-472-260	0.504
0.3	325-482-193	0.566
0.5	318-500-182	0.568
0.7	278-524-198	0.540
1.5	240-487-273	0.484

きすぎても小さすぎてもいけないという結論に至った．そのため，提案手法 MSUCB の UCB1 値計算には，暫定的に c＝0.5 を利用することにする．

6.5 効用値による影響の調査

4 つ目の実験として，シミュレート中の効用値を変化させることで，どのように AI の性能が変化するかを調査した．これまでの AI では，Playout の結果が勝利なら 1，引分けなら 0.5，負けなら 0 の効用値を得るようにしていた. 効用値による影響を調査するため, プレイヤ V, H 共に MSUCB1.5 とし, V 側の引分けの効用値を変化させながら 6×6 ボードにて対戦させた．なお，Playout 数はそれぞれ 10,000 とし, UCB1 パラメータ c は 0.5 としている．また，引分けの効用値のことを U（Draw）と表す．まず, 大まかな傾向を調べるために, U（Draw）＝0/0.25/0.5/0.75/1 の 5 通りについて 1,000 試合ずつ対戦を行った．その結果を Table 5 に示す.

Table 5 効用による影響調査(1)

U(D)	V-D-H	gain
0	44-228-728	0.158
0.25	49-796-155	0.447
0.5	57-890-53	0.502
0.75	61-924-15	0.523
1	16-978-6	0.505

結果を見ると，U（Draw）を高くするほど引分けになる確率が上昇していることが分かる．逆に，U（Draw）が低くなるほど引分けになる確率が減るが, 勝利になる確率は増えず結果として利得が減少していることが分かる．

また，U（Draw）＝U（Win）のとき，かなり高い確率で引分けになっており, 効用が上手く影響していることが確認できる．

次に，AI の性能として有望そうな U（Draw）＝0.5～1 についてより正確な影響を調査するため, 同様に V 側の U（Draw） を変化させながら 10,000 試合ずつ対戦させた.

Table 6 効用による影響調査(2)

U(D)	V-D-H	gain
0.5	575-8848-577	0.500
0.6	586-9122-292	0.515
0.7	573-9238-189	0.519
0.75	538-9318-144	0.520
0.8	561-9325-114	0.522
0.9	577-9361-62	0.526
0.95	546-9401-53	0.525
0.99	489-9463-48	0.522
1	126-9837-37	0.504

その結果を Table 6 に示す．結果を見ると，U（Draw）を上昇させることで引分け率の上昇と敗北率の減少が確認できる．また，敗北率が減少する一方で，勝率はほぼ減少することがなく，結果として利得が上昇していることが確認できる．ただし，U（Draw）＝U（Win）になると勝率も大幅に減少し，結果として利得は減少した．

これらのことから，AI の勝率を上げたいのであれば，U（Draw）をU（Win）に近づきすぎない程度に高くし，引分け率を上げたいのであれば，U(Draw)をU（Win）と同等以上にすれば良いと考えられる．

6.6 単純な UCB 探索との比較

5つ目の実験として，提案手法である MSUCB と，UCB1 値から決定的な探索を行う SimpleUCB0.5，SimpleUCB1 とを比較する実験を行う．

6.6.1 6×6 同時進行 Triomineering での比較

初めに，これまでの実験と同様に 6×6 の同時進行 Triomineering にて，プレイヤ V を MSUCB1.5，プレイヤ H を SimpleUCB0.5/SimpleUCB1 として 1,000 試合ずつ対戦させた．なお， いずれの AI も UCB1 パラメータ c は 0.5，効用は W－D－L＝1－0.5－0 としている．

まず，自分の手の UCB1 値のみを計算して考慮する SimpleUCB0.5 と 1,000 試合の対戦を行なった．なお， MSUCB1.5 は Playout 数 ＝ 10,000，SimpleUCB0.5 の Playout 数＝100,000 とした．結果は，V-D-H＝516-464-20 となり，MSUCB1.5 が大幅に勝ち越す結果となった．これは，MC と同様に相手の良い手を考慮せずに探索を行うことと，最終的な着手が決定方法の違いであると考えられる．

次に，自分の手と相手の手のそれぞれの UCB1 値を計算し，それぞれにとって UCB1 値が最大となるノードを探索する SimpleUCB1 と 400 試合の対戦を行なった．なお，MSUCB1.5 は Playout 数＝20,000，SimpleUCB1 は Playout 数＝20,000/100,000 とした．実験の結果を Table 7 に示す．

Table 7 6×6 ボードでの MSUCB1 vs SimpleUCB1

V＼H	SimpleUCB1	
MSUCB1	20,000	100,000
20,000	36-354-10	0-387-13

V-D-H

結果を見ると，Playout 数が同数の場合は少数の差は誤差の範囲であり，どちらが優位であるとも言えない結果となった．一方，SimpleUCB1 の Playout 数＝100,000 の時も少数の差は誤差の範囲内であるが，MSUCB1 の少数が 0 であるため，SimpleUCB1 の方が優位な結果となるのではないかと推測している．また，Playout 数は SimpleUCB1 が 5 倍になっているが，計算時間はおおよそ同等の時間を使用しているため，提案手法である MSUCB1 が優位であるとは言えない結果となった．

また，当初は同時進行において SimpleUCB1 などの決定的な探索では効率的な探索は行えないと予測していたにもかかわらず，SimpleUCB1 は提案手法である MSUCB1 と同等かそれ以上の性能を発揮した．そこで，SimpleUCB1 がどの様な探索を行っているかを調査するため，4×4 ボードにおいてプレイヤ V を SimpleUCB1 とし，Playout 数＝10,000 で初手の探索回数を示したのが Table 8 の（上）である．

Table8 4x4 ボードの初手の MSUCB1 探索

（上）c＝0.5 （下）c＝1

V＼H	(0, 0)	(0, 1)	(1, 0)	(1, 2)	(2, 0)	(2, 1)	(3, 0)	(3, 1)	sum
(0, 0)	1	1	8	4	4	6	1	1	26
(0, 1)	22	19	2511	2449	2448	2491	17	17	9974
sum	23	20	2519	2453	2452	2497	18	18	10000

V＼H	(0, 0)	(0, 1)	(1, 0)	(1, 2)	(2, 0)	(2, 1)	(3, 0)	(3, 1)	sum
(0, 0)	9	2	20	26	19	20	2	14	112
(0, 1)	87	73	2380	2469	2350	2412	56	61	9888
sum	96	75	2400	2495	2369	2432	58	75	10000

結果を見ると，決定的な探索を行っている SimpleUCB1 でも，MSUCB 同様に互いにとって有望な”手の組”へと探索が集中していることが分かる．MSUCB と比べると有望でない”手の組”への探索が極端に少なかったが，Table 8 （下）に示すように UCB1 パラメータ c を大きくすることで，有望ではない”手の組”へと探索を割り振ることが出来ると分かった．

また，これまでは決定的な着手は同時進行ゲーム

には適さないと論じてきたが，SimpleUCB1 は最終的な着手選択を自身の探索回数最大の手へと決定的な選択を行っているにもかかわらず，MSUCB1.5 と同等かそれ以上の結果を示した．

この原因は，6×6 の同時進行 Triomineering は互いが最善の着手を選んだ場合必ず引分けになることから，”相手の着手によって不利になる可能性のある手”を選択する必要がないからであると考えられる．このことから，6×6 のインスタンスは MSUCB1.5 と SimpleUCB1 を比較するのに適したインスタンスではなかったと考えられる．

6.6.2 G=VD の同時進行 Triomineering での比較

Cincotti らは，同時進行 Triomineering において，互いに最善の着手を選択した場合にゲームの結果が V の勝利もしくは引分けになるインスタンスのことを G=VD と定義している[3]．そこで，プレイヤ V を MSUCB1.5，プレイヤ H を SimpleUCB1 とし，G=VD のインスタンスである 8×4 ボードで 400 試合対戦させることで，最終的な着手の決定方法による差が生じるかを調査した．

実験の結果，V-D-H＝198-202-0 となり，互角の結果となった．説明は省略するが，このインスタンスは，初手の組み合わせによりゲームの結果が V か D に 1：1 の確率で決定されるが，SimpleUCB1 はプレイアウトによる利得のランダム性からか，複数回の試合を行う過程で結果的に確率的な着手を行ったことになり，最終的な着手が確率的である MSUCB1.5 と同等の性能を発揮したと推測できる．

また，詳しくは省略するが，ジャンケンをベースにしたカードゲームのシミュレータにおいて，ナッシュ均衡となる混合戦略が五分五分でない（1/3, 2/3 など確率に偏りがある）インスタンスにおいても，SimpleUCB1 は良い性能を発揮し，MSUCB1.5 の優位性を示すことはできなかった．

6.6.3 非零和同時進行ゲームでの比較

これまでの実験から，二人零和同時進行ゲームにおいて，SimpleUCB1 探索は互いのプレイヤが自分にとって有望な手への探索の割合を多くすることから，結果的にナッシュ均衡となる混合戦略と同じ様な探索を行うことを発見した．しかし，この様な互いがそれぞれにとって有望な手を決定的に選択する探索は，Table 9 に示す様な非零和同時進行ゲームにおける囚人のジレンマ的なインスタンスにおいては効率的な探索ができないと推測し，調査のための実験を行った．

Table 9 囚人のジレンマ

A＼B	b1	b2
a1	(0.5, 0.5)	(0, 1)
a2	(1, 0)	(0.2, 0.2)

これまでの実験で使用してきた同時進行 Triomineering のシミュレータは零和ゲームであり，Table 9 の様なインスタンスを作ることが困難であったため，標本平均利得が入力した表となる様にランダムで利得を返す利得シミュレータを作成し，そのシミュレータ上で非零和の囚人のジレンマ的なインスタンスにおいて MSUCB1 と SimpleUCB1 がどの様な探索を行うのかを調査した．

Table 10 囚人のジレンマにおける SimpleUCB1 と MSUCB1

SimpleUCB1

Playout数=10,000

A＼B	b1	b2
a1	3331	931
a2	841	4907

MSUCB1

Playout数=10,000

A＼B	b1	b2
a1	8	91
a2	96	9805

Playout数=100,000

A＼B	b1	b2
a1	93321	931
a2	841	4907

Playout数=100,000

A＼B	b1	b2
a1	12	136
a2	136	99716

結果を Table 10 に示す．左側が SimpleUCB1 の探索回数，右側が MSUCB1 の探索回数，上側が Playout 数＝10,000，下側が Playout 数＝100,000 である．このインスタンスにおいて，ナッシュ均衡解は（a2, b2）の組である．結果を見てみると，SimpleUCB1 はある程度（a2, b2）を探索した後に（a1, b1）へと探索箇所を変え，それ以降は（a1, b1）ばかりを探索するようになってしまうことが分かった．一方で，提案手法である MSUCB1 は，ナッシュ均衡解である（a2, b2）に探索が集中しつつも，それ以外の”手の組”にも探索が割り振られていることが確認できる．

この結果から，SimpleUCB1 と MSUCB は必ずしも同じような探索をするわけではなく，インスタンスと目的によっては MSUCB の方が適している場合もあることが示された．上記の囚人のジレンマ的なインスタンスを例に挙げると，協力ゲームであれば SimpleUCB1 が，非協力ゲームであれば MSUCB1 が適していると考えられる．

7. まとめ

本稿では，二人同時進行ゲームにおける AI 開発の足掛かりとして, ナッシュ均衡となる混合戦略を利用したモンテカルロ木探索として Mixed Strategy applied UCB 探索（MSUCB）を提案した．提案した手法は二人零和同時進行ゲームである同時進行 Triomineering において，原始的なモンテカルロ法（MC）に対して優位な性能を発揮した．また，MC の問題点であった相手の良い着手を考慮しないという点を解決することに成功した.

さらに，提案手法である MSUCB に関して，探索深さを拡張することで性能を向上させることが出来る可能性を示した.

また，UCB1 値計算の際のパラメータ c や，プレイアウトにおける効用値などの影響も調査し, パラメータによる性能の変化の傾向を示した.

一方で, 提案手法は二人零和同時進行ゲームである同時進行 Triomineering においては，従来手法である単純な UCB 探索を組み合わせた探索（SimpleUCB1）に対して，優位と言える性能を発揮することは出来なかった．そこで，SimpleUCB１に関しての調査を行ったところ, 零和ゲームにおいては確率的なノード探索を行なわなくても, 結果的に提案手法による探索と同様の探索を行なうことが出来ることを発見した.

また一方で, 非零和ゲームにおける囚人のジレンマにおいては，提案手法である MSUCB と従来手法である SimpleUCB１とで差異が生まれることを示し, 対象となるインスタンスや目的によって, 提案手法である MSUCB が有効となる場合が存在することを示した.

今後の課題として第一に挙がるのは, 現在はナッシュ均衡の近似アルゴリズムをある種の力技で計算する手法を取っているが, この手法にはナッシュ均衡に近似されるという数学的な裏付けがなく, また近似に必要となる計算時間が多いため, 数学的な証明や実績のある Counterfactual Regret minimization[5]などのアルゴリズムを実装し，ナッシュ均衡となる混合戦略の近似速度と精度を向上させることで, 手法全体の性能を向上させることが出来ると考えられる.

また, 提案手法は現時点ではあらかじめ探索深さを固定しているが，提案手法は 6．3 章にて示したように深さの拡張による性能向上が見込めるため, 通常の UCT 探索の様にノードの探索回数によって木を成長させる MSUCT を実装することで, より高い性能を発揮することが出来ると考えられる.

また, 現時点での提案手法 MCUCB に関しても, Playout 数を増やすことによる性能の伸び方を調査することで, より正確に他の手法との比較を行うことや, より行動数や状態数の多いゲームにおいても提案手法が有効かどうかを調査することで, 手法そのものの一般性を測る必要がある.

参考文献

[1] 小森 成貴, Alessandro Cincotti, 飯田 弘之, “同時着手 Domineering におけるモンテカルロ法の適用”, IPSJ GI, 2008(59), 51-58, 2008

[2] Alessandro Cincotti, Komori Shigetaka, Hiroyuki Iida, “The Game of Synchronized Triomineering and Synchronized Tridomineering”, International Journal of Computational and Mathematical Sciences, 2(3): 143-148, 2008

[3] Tao Cao, Alessandro Cincotti and Hiroyuki Iida, “New Results for Synchronized Triomineering”, International MultiConference of Engineers and Computer Scientists, 2012, Vol II, 2012

[4] S.A. BLANCO, and AVIEZRI S. FRAENKEL, “TRIOMINEERING, TRIDOMINEERING AND L-TRIDOMINEERING”, Technical Report , MCS04-10, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science

[5] Martin Zinkevich, Michael Johanson, Michael Bowling, Carmelo Piccione, “Regret Minimization in Games with Incomplete Information”, Advances in Neural Information Processing Systems, Vol.20, 99. 1729-1736, 2008

[6] L. Kocsis, C. Szepesvari, “Bandit based Monte-Carlo Planning”, LNCS vol.4212 (ECML 2006), pp. 282-293, 2006

[7] 田中 嘉浩, “Nash 均衡の計算複雑度について”, Economic Studies, 59(4): 9-15, 2010

[8] Sylvain Gelly, Yizao Wang, Remi Munos, Olivier Teytaud, “Modification of UCT with Patterns in Monte-Carlo Go”, Technical Report 6062, INRIA

[9] 岡田 章（1996）, “ゲーム理論”, 有斐閣