Vol.2021-GI-45 No.12
2021/3/5

コンピュータ大貧民における各種モンテカルロ法の検討

富岡 聖[1]　大久保 誠也[1]　湯瀬 裕昭[1]　武藤 伸明[1]

概要：本研究の目的は強い大貧民 AI の実現である. そこで 2 つのモンテカルロアルゴリズムを提案する. 1 つ目は 1 ゲームの開始前にそのゲーム中, どのバンディットアルゴリズムを用いるかを動的に変更する手法である. また, この変更もバンディットアルゴリズムにより行う. 2 つ目は相手プレイヤが行うモンテカルロシミュレーションの精度を下げるようにプレイする手法である. 提案手法の有効性を検証するため, Blauweregen をベースとしたプログラムを作成し, いくつかの計算機実験を行った. 計算機実験の結果から, 統計的に有意とまではいえないものの, ある程度の有効性が明らかになった.

キーワード：コンピュータ大貧民, モンテカルロ法, バンディットアルゴリズム

1. はじめに

大貧民は多人数不完全情報ゲームの一つであり, 大貧民をコンピュータ上で対戦させるのが, コンピュータ大貧民である. このとき, プレイヤに相当するプログラムを, クライアントプログラムもしくは大貧民 AI とよぶ. 毎年電気通信大学にて, 大貧民 AI の強さを競う, UEC コンピュータ大貧民大会 (UECda) が開催されている. 機械学習を用いるプログラムが想定されている無差別級では, モンテカルロシミュレーション (モンテカルロ法) を用いたプログラムが良い成績を収めている.

2016 年度 UECda 無差別級の優勝プログラムである大渡の Blauweregen は, 大貧民の知識を用いた方策関数を使用しており, そのパラメータは方策勾配法による強化学習により設定されている [10]. また, このプログラムではモンテカルロ法のバンディットアルゴリズムに UCB-root を用いている. 他にも, UECda 無差別級で優秀な成績を収めているプログラムには, モンテカルロ法を用いているものが多い.

また, コンピュータ大貧民におけるレーティング手法に関する研究も行われている. 五ケ谷らは大貧民特有の性質を考慮したレーティング手法についての研究を行い. その結果, プレイヤの間に相性関係があることを明らかにした [7]. ここで, 相性関係とは, プレイヤ間の強弱が, 全順序関係や半順序関係とはならないことをいう.

このように, モンテカルロ法は, 大貧民の研究において重要な役割を果たしている. しかしながら, どのモンテカルロ法がもっとも有効かは明らかではない. また, モンテカルロ法の手法に相性があるかや, モンテカルロ法プログラムに対して有利に戦える戦法も明らかになっていない.

本研究の目的は, 強い大貧民 AI を実現することである. そこで 2 つのモンテカルロアルゴリズムを提案する. 1 つ目は選択バンディット手法である. これは, あるゲームを開始する前に, そのゲームで使用するバンディットアルゴリズムを選択する. この選択自体も, バンディットアルゴリズムを用いて行う. 2 つ目は敵シミュレーションの精度を下げるプレイング手法である. 相手が行うモンテカルロシミュレーションの精度が下がるように自分の提出手を選択する. 提案手法の有効性を検証するため, 計算機実験を行った. 具体的には, Blauweregen をベースとしていくつかのプログラムを作成し, それぞれのアルゴリズムの挙動の違いや強さの違いを検証した.

2. コンピュータ大貧民

大貧民は, トランプカードを用いた日本発祥のゲームである. 複数人でプレイされ, 相手の手札が未知であるため多人数不完全情報ゲームに分類される. また大貧民にはさまざまなローカルルールが存在する.

コンピュータ大貧民は大貧民を計算機上で行うゲームである. 本論文では大貧民を計算機上でプレイさせるプログラムを大貧民 AI とよぶ. 2006 年から電気通信大学で UEC コンピュータ大貧民大会 (UECda) が開催されている. この大会は持ち寄った大貧民 AI を対戦させ順位を競う大会である. 大貧民の研究の多くは UECda のルールに準じて行われている. 本研究もこの大会のレギュレーションに沿っ

[1] 静岡県立大学大学院経営情報イノベーション研究科

ている. UECda で使用される UEC 標準ルールは以下のようなものである.

ゲームの単位　下記の手札配布, カード交換, カードの提出を経て, プレイヤ 1 人を残した他の全プレイヤが上がるまでが 1 ゲームである. 大会では複数のゲームを繰り返し行うことで最終的な順位を争う.

階級　最初にすべての手札を提出し終えたプレイヤから順に, 大富豪, 富豪, 平民, 貧民, 大貧民という序列付けを行う.

手札配布　初回はランダムに席順 (カードを出す順番) を決定し, ランダムに各プレイヤに配布する. 大貧民, 大富豪などの序列が決定される 2 回目以降は, 大富豪を起点として席順にカードを配布する.

カード交換　ゲーム開始前に, 直前のゲームの結果を基に, カード交換を行う. 大富豪は大貧民に任意のカードを 2 枚渡し, 大貧民は大富豪に強いカードから順に 2 枚渡す. 富豪は貧民に任意のカードを 1 枚渡し, 貧民は富豪にもっとも強いカードを 1 枚渡す. 平民はカード交換を行わない.

カードの提出　順番が回ってきたプレイヤは, カードを出すか, パスを選択できる. 場に何もない状態であれば, 好きなカードを出すことができる. 場にカードが出ている場合は, 場のカードと同じ役で, 場のカードよりも強いカードを出すことができる. 役が階段の場合は, 出すカードすべてが場にある階段のカードすべてより強くなければいけない. また, 提出可能な手のことを合法手とよぶ.

3. モンテカルロ法

3.1 モンテカルロ法とは

モンテカルロ法とは, 乱数を用いてシミュレーションを行う手法の総称である. ゲーム情報学においては, 各選択肢の評価値を求めるために使用されている. 大貧民における素朴なモンテカルロ法のアルゴリズムは以下のように行われる.

(1) 自分の手札と場に出ているカードの情報から, 出せる手をすべて列挙する.
(2) 以下を, 設定した回数, 繰り返す.
  (a) すべてのカードから自分の手札とすでに場に出ているカードを除いたカードからランダムに相手に配布する.
  (b) (1) で求めた手のなかから 1 つの手 i をランダムに選択する.
  (c) 選んだ手 i を提出した次の局面から, 終局までランダムにシミュレートする.
  (d) 終局時の順位から手 i の評価値 $\overline{x_i}$ を更新する.
(3) 評価値がもっとも高い手 i を提出する.

ここで, (2)(c) のシミュレーション部分を, プレイアウトとよぶ. また, (2)(b) においてシミュレーションを行う手を選択する際, バンディットアルゴリズムが使用されることが多い. 代表的なアルゴリズムとしては, 以下のようなものがある.

UCB1-Tuned　UCB1-Tuned[6] は UCB1[2] の第 2 項を改変し, 分散も考慮する手法である. $\hat{\sigma}_i$ を取得した報酬の分散とすると,

$$V_i = \hat{\sigma}_i^2 + \sqrt{\frac{2\log(s)}{n_i}}$$

として

$$\arg\max_{i \in A} \left(\hat{\mu}_i + \sqrt{\frac{\log(s)}{n_i} min(\frac{1}{4}, V_i)} \right)$$

のアームを選択する. UCB1-Tuned を用いたプログラムは 2019 年の UECda 無差別級において報告されている. また, snowl[9] が代表的なプログラムとして知られている.

UCB-root　UCB-root はパラメータを c として,

$$\arg\max_{i \in A} \left(\hat{\mu}_i + c\sqrt{\frac{s}{n_i}} \right)$$

のアームを選択する. 大渡 [10] の Blauweregen が UCB-root を用いている.

UCB-V　UCB-V[4] はより分散を考慮した手法である. $\zeta > 0$, $c > 0$, b を定数すると.

$$\arg\max_{i \in A} \left(\hat{x}_i + \sqrt{\frac{2\hat{\sigma}_i E(s)}{n_i}} + c\frac{3bE(s)}{n_i} \right)$$

のアームを選択する. ここで,

$$E(t) = \zeta \log(t)$$

である. UCB-V を使用したプログラムは過去の UECda において報告されていない.

他にも, UECda で使用されたことがあるアルゴリズムとしては Thompson Sampling[1], [5], 使用されたことがないアルゴリズムとしては, KL-UCB[3], UCB2[2] などがある.

3.2 Blauweregen

2017 年度 UECda 無差別級の優勝クライアントである. 大渡によって開発された [10]. Blauweregen は方策関数を使用しており, そのパラメータは方策勾配法による強化学習により設定されている. モンテカルロシミュレーションの際には, 相手の手札推定を行い, その後, UCB-root を用いたモンテカルロシミュレーションを行う. また, UCB-root により手を選んだ後は, 方策関数を用いたシミュレーショ

ンを行っている. Blauweregen のモンテカルロシミュレーション中の簡単な挙動を, 以下に示す.

(1) 相手の手札推定を行い, 手札配置を複数個作成.
(2) 以下を, 設定した回数, 繰り返す.
  (a) UCB-root により, シミュレーションを行う手 i を選ぶ.
  (b) 複数個の手札配置から一つ選ぶ.
  (c) (b) で選んだ手札配置を用い, 方策関数により, (a) で選んだ手 i についてのプレイアウトを行う.
  (d) 手 i の評価値を更新する.
(3) 評価値がもっとも高い手 i を提出する.

Blauweregen は過去の UECda 参加クライアントのなかでも最強クラスの実力があり, これより明確に強いプログラムは報告されていない.

4. 提案手法 1 選択バンディット

大貧民大会においては, さまざまなバンディットアルゴリズムが用いられている. 大貧民にもっとも適したバンディットアルゴリズムが存在するならば, そのアルゴリズムを用いることが, もっとも強さに結びつくと考えられる.

Blauweregen ならびにそこから派生した tommy と Glicine の間には, 学習パラメータとバンディット手法以外に大きな違いが無い. それにも関わらず, 3 すくみの関係が存在している [7]. このことは, 適したバンディットアルゴリズム手法は 1 つに定まらず, 相性関係があることを示唆している. そこで, バンディットアルゴリズムを試合中に動的に変更する, 選択バンディットアルゴリズムを提案する.

選択バンディットアルゴリズムは, 各ゲームの開始時に, どのバンディット手法を用いるかを選択する. この選択自体も, バンディット手法を用いて行う. コンピュータ大貧民大会は数千ゲームから 1 万ゲームで, 1 試合が構成される. ゲームが進むほど, その試合で有効なバンディットアルゴリズムに収束することが期待できる.

選択バンディットアルゴリズムは, 以下のような流れで実行される.

(1) 以下を, 1 試合が終わるまで繰り返す.
  (a) ゲーム開始時に, バンディットアルゴリズムに基づいて, どのバンディット手法を用いるのかを選択する.
  (b) (a) で選択したバンディットアルゴリズムを用いて, 1 ゲームを行う.
  (c) ゲームの結果から, バンディットアルゴリズムの評価値を決める.

“選択するバンディットアルゴリズム (選択バンディットアルゴリズム)” ならびに “選択バンディットアルゴリズムで選択されるバンディットアルゴリズム (被選択バンディットアルゴリズム)” として何を使うかは, 任意である. どのアルゴリズムを選択するかや, 各パラメータをどのようにするかは, 別途決定する必要がある.

Algorithm 1 Choice Bandit

```
games ← 0
while games < AllGame do
  for bandit ∈ Bandit do
    v_bandit ← funcChoiceBandit(bandit)
  end for
  b ← arg max_{bandit∈Bandit} (v_bandit)
  while one game not finished do
    play a game with b
  end while
  update results {Update probability distribution, etc.}
  games ← games + 1
end while
```

選択バンディットを使用して全体の試合を進める疑似コードを Algorithm1 に示す. ここで, Algorithm1 内の各識別子の意味は以下の通りである.

AllGame : 全試合数.

Bandit : 使用するバンディットアルゴリズムの集合.

funcChoiceBandit : 引数 bandit を受け取り, 選択バンディットで使用するバンディットアルゴリズムにより bandit の評価値を返す関数.

5. 提案手法 2 敵シミュレーションの精度を下げるプレイング手法

不完全情報ゲームである大貧民において, 精度の高いモンテカルロシミュレーションを行うことは, 強さに結びつくと考えられる. 一方, モンテカルロシミュレーションの精度を決定する要因としては, 性能の良いバンディットアルゴリズムを使用することや, 高い精度で相手手札を推定すること, モンテカルロ木の複雑さ等が考えられる. なお, ここでモンテカルロ木の複雑さとは, 探索木の大きさのことをさすものとする. これまで精度の高いモンテカルロシミュレーションを行うための研究や, 敵プレイヤの得点を下げる研究 [8] は行われてきたが, 相手のシミュレーションの精度を下げるための研究は行われてきていない.

そこで, 相手プレイヤがモンテカルロシミュレーションを試合中に行っていることを前提とし, 相手プレイヤのモンテカルロシミュレーションの精度を下げるアルゴリズムを提案する. 相手もモンテカルロシミュレーションを行っているのならば, 第 3 節で示したように, 列挙した各合法手に対して, 設定した回数のシミュレーションを割り振っているはずである. そして, 合法手数が多いほど, 各合法手に割り振ることのできるシミュレーション回数は減少し, シ

ミュレーションの精度は下がる. つまり, 相手の合法手が多くなる手を提出することにより, 相対的に自らの強さをあげることができる.

提案手法は, 各合法手を提出した後の探索木の大きさを計算する. そして, より木が大きくなる手を出しやすくなるように, 評価値を調整する. 具体的には, プレイアウト中に到達したノードの葉ノードの合計を求め, その平均値を評価値に加味する. 以下ではこれを DESA 値 (Decrease Enemy's Simulation Accuracy) とする.

1 回のプレイアウト中に DESA 値を求めるアルゴリズムは, 以下のような流れで実行される. ここで, 変数 s はモンテカルロシミュレーション開始時, 0 に初期化されているものとする.

(1) バンディットアルゴリズム等により, シミュレーションを行う手を選択する.
(2) 相手にカードを割り振る.
(3) 以下を, (1) で選択した手を提出した状態から終局まで, 繰り返す.
  (a) 順番が回ってきたプレイヤの合法手をすべて挙げ, その個数を s に加算する.
  (b) 合法手から選んだ手を提出し, 次の状態に盤面を進める.
(4) プレイアウト終了時の s の平均値をシミュレーションを行った手の DESA 値とする.

DESA 値を求めるアルゴリズムを表した疑似コードを Algorithm2 に示す. ここで, Algorithm2 内の各識別子の意味は以下の通りである.

M_t^p : 時刻 t におけるプレイヤ p の合法手の集合. ただし $0 \leqq p \leqq 4$ であり, $p = 0$ を自プレイヤとする. また, $t = 0$ をシミュレーション開始時刻とする.

AllSimulations : 全シミュレーション回数.

n_i : 合法手 i の合計シミュレーション回数.

$leafSum_i$: 合法手 i の葉ノードの合計.

求めた DESA 値は, 評価値に加算される. 本研究では, 加算の方法により, 2 つの手法を提案する. 1 つ目は, 各モンテカルロシミュレーション中の各合法手のバンディット値に DESA 値を加算する手法である. 2 つ目は, すべてのモンテカルロシミュレーション終了後の各合法手の評価値 $\overline{x_i}$ に DESA 値を加算する手法である. 前者は, シミュレーションを行う手の選択に影響を与えるが, 後者はシミュレーションを行う手の選択には影響を与えない.

Algorithm 2 Decrease enemy simulation accuracy

Ensure: $DESA_i$
$H \leftarrow M_0^0$
$simulations \leftarrow 0$
$t \leftarrow 0$
$L \leftarrow 0$
$leafSum_i \leftarrow 0$ $(i = 0...|H|)$
$m_i \leftarrow 0$ $(i = 0...|H|)$
$DESA_i \leftarrow 0$ $(i = 0...|H|)$
while $simulations < AllSimulations$ **do**
  Select $m \in H$ *by bandit algorithm*
  while Playout is not finished **do**
    $t \leftarrow t + 1$
    $leafSum_m \leftarrow leafSum_m + |M_t^p|$
  end while
  update results {Update probability distribution, etc.}
  $simulations \leftarrow simulations + 1$
end while
for $i = 0...|H|$ **do**
  $m_i \leftarrow leafSum_i / n_i$
end for
$L \leftarrow \sum_{i=0}^{|H|} m_i$
for $i = 0...|H|$ **do**
  $DESA_i \leftarrow m_i / L$
end for

6. 計算機実験 1 選択バンディットの有効性の評価

6.1 実装プログラム Ninja

提案手法 1 を実装したプログラム Ninja を作成した. まず, 各バンディットアルゴリズムの選定である. 本研究では, 選択バンディットアルゴリズムには UCB-V を使用した. 以下ではこれを選択 UCB-V とする. 選択されるバンディットアルゴリズムには 6 種類のバンディットアルゴリズムを実装した. その内訳は UCB-root, UCB1-Tuned, Thompson Sampling, KL-UCB, UCB-V, UCB2 である. 各バンディットアルゴリズムの実装の際, UCB-root のパラメータは, $c = 0.7$, UCB-V の各パラメータは, $\zeta = 0.5$, $c = 2$, $b = 1$ としている. 選択 UCB-V の実装の際, 各バンディットアルゴリズムの確率分布にはベータ分布を使用している. また, ベータ分布の更新作業は Algorithm1 の update results によって行っており, 報酬値は土橋 [11] のモンテカルロ法における報酬値の実験結果より, 通常の報酬値の 2 乗値を用いている.

6.2 各バンディットアルゴリズムの強さの検証

被バンディットアルゴリズムを決める必要があるが, 明らかに弱いアルゴリズムを含める必要はない. そこで, 各バンディットアルゴリズム単体の強さを検証するため, 試合中に単一のバンディットアルゴリズムのみを使用するクライアント同士の対戦を行った. 具体的には, Blauweregen のプログラムにおいて, UCB-root を使用する部分のみを各バンディットアルゴリズムに置き換えたプログラム同士を対戦させた.

表 1 各組み合わせに対して, 各バンディットアルゴリズムが 1 位になった試合数

	No.1	No.2	No.3	No.4	No.5	No.6
UCB-root	20	19	29	24	26	—
UCB1-Tuned	29	35	25	26	—	23
Thompson Sampling	16	21	20	—	25	20
UCB-V	20	25	—	24	31	36
KL-UCB	15	—	26	26	18	21
UCB2	—	0	0	0	0	0

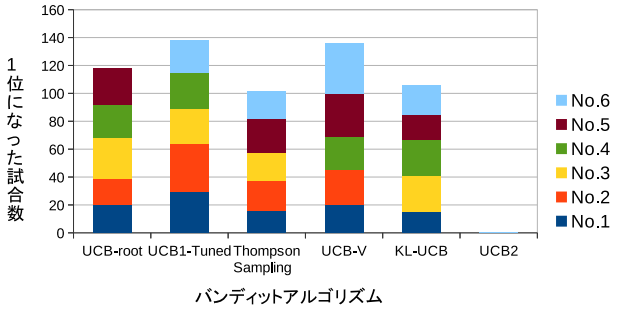

図 1 各バンディットアルゴリズムの 1 位になった試合数の結果

実験では UCB-root, UCB1-Tuned, Thompson Sampling, UCB-V, KL-UCB, UCB2 をそれぞれ実装したクライアントから, 5 つを選んできて対戦を行わせた. なお実験は, 1 試合 3000 ゲームとし, 各組み合わせに対して 100 試合ずつ対戦を行った. 各バンディットアルゴリズムの評価は, 100 試合で 1 位になった回数, 平均得点で行う.

試合の組み合わせと, 各組み合わせにおいて各バンディットアルゴリズムが 1 位になった試合数を表 1, 図 1 に示す. また, 各バンディットアルゴリズムの平均得点を図 2 に示す.

表 1, 図 1 より UCB2 は 1 試合も勝てていない. また図 2 より UCB2 の平均得点は 600 点差以上の差で, 他のバンディットアルゴリズムに比べて低い結果となった. さらに, UCB2 以外のプログラム同士の平均得点の差は, どの組み合わせにおいても 100 点以内という結果となった. 以上の結果より, UCB2 以外のバンディットアルゴリズム間には強さの違いがほとんどないことがわかる.

これらの結果を踏まえ, これ以降, Ninja の被選択バンデットアルゴリズムには UCB2 以外の 5 つを使用するものとする.

6.3 選択バンディットアルゴリズムの収束確認

提案手法は, 対戦相手によって最適なアルゴリズムは異なることを前提としている. そこで, 選択 UCB-V は, 対戦相手が異なれば, 異なるバンディットアルゴリズムを選択しているのかを明らかにするため, 計算機実験を行った. 具体的には, いくつかのライト級クライアントと対戦させ, 選択されたバンディット手法に差があるか否かを検証した. ライト級のプログラムは機械学習を用いないプログラムが

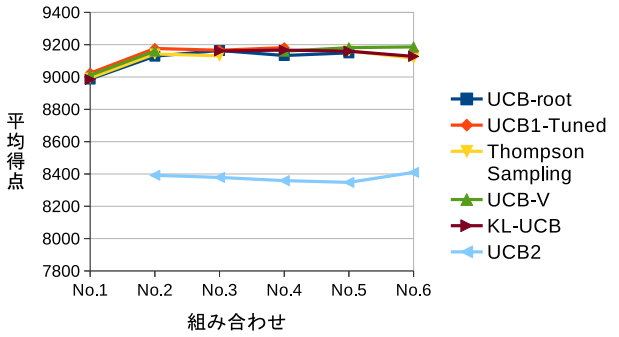

図 2 組み合わせ毎の各バンディットアルゴリズムの平均得点の結果

想定されており, 試合中にプレイスタイルが大きく変化するようなことがないと考えられる. そのため, ライト級のクライアントプログラムと対戦させ, 各バンディットアルゴリズムが選択された平均回数の間に差が存在するならば, 選択 UCB-V は対戦相手に適したバンディットアルゴリズムを多く選択しているといえる. 実験で使用するプログラムは過去のライト級優勝プログラムから選択した. 試合の組み合わせは, 次の 3 つである.

(1) Ninja, kou2, kou2, kou2, kou2
(2) Ninja, st-kou, st-kou, st-kou, st-kou
(3) Ninja, VV-8, VV-8, VV-8, VV-8

これ以降, 本研究で作成した以外の, 対戦相手として出てくる各プログラムは, UEC コンピュータ大貧民サイトで公開されているものを用いている. また, サーバープログラムは tndhm_devkit_c-20180826 に同梱されているものを用いた. 計算機は 3 台使用し, CPU はそれぞれ AMD Ryzen 5 3500 6-Core Processor 3.59GHz, Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz LGA1150, Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz (2200.05-MHz K8-class CPU) である.

対戦相手によって最適なバンディットアルゴリズムに収束したかどうかの検証は, 115 試合の各バンディットアルゴリズムの平均使用率で行う. また, UECda では 1 試合中に含まれるゲーム数が年によって異なることと, 割合が収束するタイミングが不明なことから, 3000, 5000, 8000, 10000, 13000, 15000 ゲーム終了時の, 各タイミングで集計を行った.

対 kou2 との実験におけるゲーム数と各バンディットアルゴリズムの平均使用率を図 3 に, 対 st-kou との実験におけるゲーム数と各バンディットアルゴリズムの平均使用率を図 4 に, 対 VV-8 との実験におけるゲーム数と各バンディットアルゴリズムの平均使用率を図 5 に示す.

図 3 より, 対 kou2 との対戦では試合全体を通して, UCB-root, UCB1-Tuned, UCB-V が多く選択されている. 特に, UCB-root と UCB-V はゲーム数が多くなるほど多く選択されている. 図 4 より, 対 st-kou との対戦では試合全体を

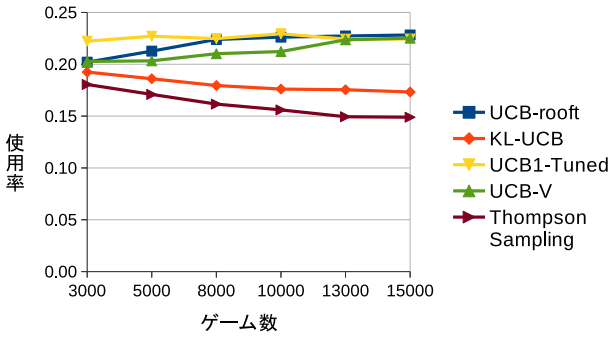

図 3 対 kou2 の実験におけるゲーム数と各バンディットアルゴリズムの使用率の推移

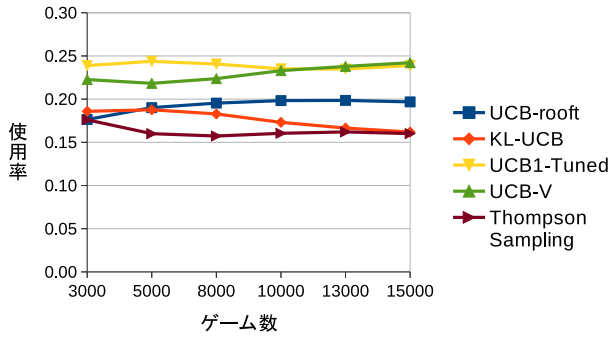

図 4 対 st-kou の実験におけるゲーム数と各バンディットアルゴリズムの使用率の推移

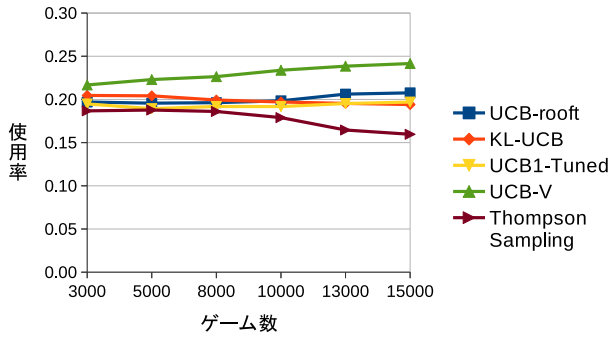

図 5 対 VV-8 の実験におけるゲーム数と各バンディットアルゴリズムの使用率の推移

通して, UCB1-Tuned, UCB-V が多く選択されている. 図 5 より, 対 VV-8 との対戦では試合全体を通して UCB-V がもっとも多く選択されている. また, 3 つの組み合わせの対戦結果を通して KL-UCB, Thompson Sampling は, もっとも多く選択されているバンディットアルゴリズムに比べて平均使用率が低い結果となった. 以上の結果より, 多く選択されているバンディットアルゴリズムは kou2, st-kou, VV-8 のそれぞれで異なるため, 選択 UCB-V は対戦相手が異なれば, 異なるバンディットアルゴリズムを多く選択していることがわかる.

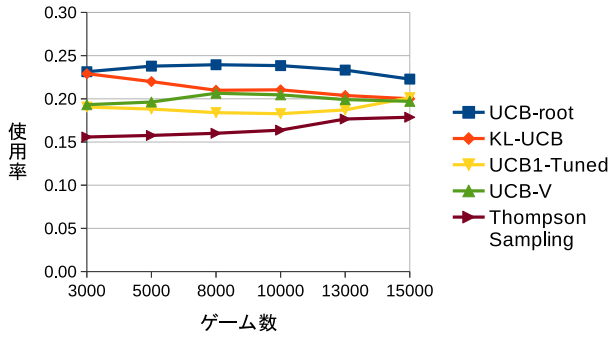

図 6 ゲーム数と各バンディットアルゴリズムの使用率の推移

6.4 選択 UCB-V プログラムの強さと有効性の検証

提案手法により, 強い大貧民 AI が実現できているかを検証するため, 計算機実験を行った. 具体的には, Ninja の対戦相手として Blauweregen とその他の任意のプログラムを 3 つ選択し, さまざまな組み合わせの試合を行った. さまざまな組み合わせで, 改良元である Blauweregen と対戦させることにより, Blauweregen よりも強くなっているかや, Blauweregen よりも相性関係に左右されない結果を出せているのかを検証した.

Ninja, Blauweregen と対戦させるプログラムは, 過去の UECda の 11 個の優勝クライアントプログラムを用いた. 具体的には, 無差別級の優勝クライアントからは tommy, Glicine, FujiGokoro, wisteria, snowl, paoon, beer-song, crow を, ライト級の優勝クライアントからは kou2, st-kou, VV-8 を用いた. 試合では, これらのプログラムからランダムに 3 つを選択した.

Ninja と Blauweregen との比較は 115 試合で, Ninja が Blauweregen に勝った試合数, Ninja が 1 位になった試合数, Blauweregen が 1 位になった試合数, 平均得点により行う. また, これらの結果と各バンディットアルゴリズムの使用率により, 相性関係の問題を解決しているかの検証を行う. UECda では 1 試合中に含まれるゲーム数が開催年度によって異なることと, 割合が収束するタイミングが不明なことから, 3000, 5000, 8000, 10000, 13000, 15000 ゲーム終了時の, 各タイミングで集計を行った.

表 2 に試合結果を示し, 表 3 に平均得点を示す. また, 図 6 に各バンディットアルゴリズムの平均使用率を示す.

表 2 より Ninja が Blauweregen に勝った試合数はすべてのゲーム数において Blauweregen よりも上回っている. また, Ninja が 1 位になった試合数も Blauweregen が 1 位になった試合数と比べると全体的に上回っていることがわかる. また, 表 3 より平均得点も大きな差はみられない. しかしながら, これらの結果は検定で有意差が出るほどの差ではない. 以上のことから, このパターンにおける Ninja は Blauweregen よりも, 強くも弱くもなっているとはいえないが, 相性関係に左右されないプレイングができていると

表 2　各ゲーム数の試合結果

ゲーム数	Blauweregen に勝った試合数 (割合)		Ninja が 1 位になった試合数 (割合)		Blauweregen が 1 位になった試合数 (割合)	
3000	62	(0.53913)	58	(0.50435)	45	(0.39130)
5000	58	(0.50435)	56	(0.48696)	53	(0.46087)
8000	58	(0.50435)	55	(0.47826)	52	(0.45217)
10000	58	(0.50435)	57	(0.49565)	53	(0.46087)
13000	59	(0.51304)	53	(0.46087)	50	(0.43478)
15000	59	(0.51304)	52	(0.45217)	52	(0.45217)

表 3　各ゲーム数の Ninja と Blauweregen の平均得点

ゲーム数	Ninja	Blauweregen
3000	9,995.41	9,981.53
5000	16,656.80	16,628.23
8000	26,628.70	26,622.74
10000	33,288.23	33,242.10
13000	43,256.21	43,196.63
15000	49,880.27	49,846.17

いえる. また, 図 6 より UCB-root が平均的に多く選択されている. このことから UCB-root は Blauweregen を含めた比較的多くの対戦の組み合わせにおいて有効であることがわかる.

以上の実験結果より, 選択 UCB-V を用いることにより, 最終得点において Blauweregen より強い大貧民 AI を実現することはできないが, 被バンディットアルゴリズムに UCB-root, KL-UCB, UCB1-Tuned, UCB-V, Thompson Sampling を用いることにより, 相性関係を解決した強い大貧民 AI を実現可能であることがわかる.

7. 計算機実験 2 敵シミュレーションの精度を下げるプレイング手法の有効性の評価

7.1 実装プログラム Sinobi

提案手法 2 を実装したプログラム Sinobi を作成した. 具体的には, DESA 値をシミュレーション中に使用した Sinobi(Simulation), DESA 値をモンテカルロシミュレーション後に使用する Sinobi(Last), DESA 値をモンテカルロシミュレーション中と後の両方で使用する Sinobi(Simulation + Last) の 3 つのプログラムを実装した.

また, 提案手法 2 の実装では, バンディット値または各合法手の評価値と, DESA 値のバランスを取るため, DESA 値に 0.1 を掛ける工夫をしている. 本研究では各評価値と DESA 値のバランスに関しては研究の対象に含めていない.

7.2 実装プログラムの強さの検証

提案手法 2 の有効性を検証するために, 計算機実験を行った. Sinobi は対モンテカルロ法を想定したプログラムであるため, 対戦の組み合わせは Sinobi, Blauweregen, tommy, Glicine, Ganesa の 5 つとした. 1 試合は 3000 ゲームである. 評価は, Sinobi と, Sinobi のベースとなった Blauweregen との比較により行った. 比較は, Blauweregen に勝った試合数, Sinobi が 1 位になった試合数, Blauweregen が 1 位になった試合数により行う. なお, Sinobi(Simulation) の入った試合は 100 試合. Sinobi(Last) の入った試合は 200 試合, Sinobi(Simulation + Last) の入った試合は 100 試合, 行っている.

表 4 に DESA 値を使用した場合における各試合結果を, 表 5 に平均得点を示す.

表 4 より, DESA 値をシミュレーション中と DESA 値をシミュレーション後に使用する場合では Blauweregen に勝った試合数の割合が 0.5 を上回っている. また, Sinobi が 1 位になった試合数はすべての場合において Blauweregen が 1 位になった試合数を上回っている結果となった. しかしながら, これらの結果は検定で有意差が出るほどの差ではない. 以上の結果から DESA 値をシミュレーション後に使用する場合はモンテカルロシミュレーションを行っているプログラムに対して有効に働いていることがわかる.

8. 考察

本研究では主に, 選択バンディットアルゴリズムの有効性を検証する実験と, 相手のシミュレーションの精度を下げることを目的としたプレイングの有効性を検証する実験を行った. ここでは, 計算機実験全体をまとめるとともに, 提案手法の有効性と, 強い大貧民 AI について考察する.

第 6.4 節から, 選択 UCB-V はプレイヤ間の相性関係を解決していると考えられる. 選択バンディットアルゴリズムを用いることにより, 大貧民をプレイする強い大貧民 AI を実現するという目的は, 相性関係に左右されないプレイングを実現したという面で達成できたと考えられる. ただ, 選択バンディットアルゴリズムによってプログラム単体の強さという点においては目的を達成することはできなかった. プログラムのさらなる強化のため, 本研究で実装した以外のバンディットアルゴリズムを導入することなどが考えられる.

第 7 節より提案手法 2 はモンテカルロ法を使用したプロ

表 4 DESA を使用した場合における各試合結果

	試合数	Blauweregen に勝った試合数 (割合)		Sinobi が 1 位になった試合数 (割合)		Blauweregen が 1 位になった試合数 (割合)	
Simulation	100	55	(0.550)	41	(0.410)	35	(0.350)
Last	200	107	(0.535)	79	(0.395)	63	(0.315)
Simulation + Last	100	49	(0.490)	39	(0.390)	35	(0.350)

表 5 各ゲーム数の Sinobi と Blauweregen の平均得点

ゲーム数	Sinobi	Blauweregen
Simulation	9,367.34	9,346.40
Last	9,376.12	9,344.55
Simulation + Last	9,366.03	9,372.49

グラムに対して優位に戦えるという面で, 目的を達成できたと考えられる. 特に, DESA 値をモンテカルロシミュレーション後に使用する場合において有効であると考えられる. 一方, シミュレーション中に DESA 値を使用する場合と, シミュレーション中と後の両方で DESA 値を使用する場合において, 比較的良い結果がでなかった. これは DESA 値が収束していないシミュレーション序盤では上手くシミュレーションを割り振ることができていないことが原因の一つである可能性がある. さらにプログラムを強化するために, 手札推定の精度をより高めることや, 相手のシミュレーションの精度を下げる手と各合法手の評価値とのバランスを最適化することなどが考えられる.

今回, 2 つの手法を提案したが, それぞれを別々に実装している. 同時に使用することにより, さらなる強さの向上が期待できる. 以上より, 強い大貧民 AI の実現には, 被選択バンディットに UCB-root, KL-UCB, UCB1-Tuned, UCB-V, Thompson Sampling を用いた選択 UCB-V を使用する手法を用いるとともに, すべてのモンテカルロシミュレーション終了後の各合法手の評価値 $\overline{x_i}$ に DESA 値を加算する手法を用いると良いといえる.

9. おわりに

本研究の目的は, 強い大貧民 AI を実現することである. そこで, 2 つのモンテカルロアルゴリズムを提案した. 1 つ目は, 選択バンディット手法である. これは, あるゲームを開始する前に, そのゲームで使用するバンディットアルゴリズムを選択する. この選択自体も, バンディットアルゴリズムを用いて行う. 2 つ目は, 敵シミュレーションの精度を下げる手法である. 相手が行うモンテカルロシミュレーションの精度が下がるように, 自分の提出手を選択する.

提案手法の有効性と各バンディットアルゴリズムの有効性を明らかにするため, いくつかの計算機実験を行った. その結果, 選択 UCB-V プログラムはプレイヤ間の相性関係を解決していると考えられる結果となった. また, 相手のシミュレーションの精度を下げることを目的としたプレイングについての有効性を確認できた.

今後の課題としては, 選択バンディットアルゴリズムに UCB-V 以外のバンディットアルゴリズムを使用する場合における各モンテカルロアルゴリズムの挙動や強さの違いを検証することや, 選択バンディットアルゴリズムと敵シミュレーションの精度を下げるプレイング手法を組み合わせた場合の強さの検証などがある. また, 敵シミュレーションの精度を下げるプレイング手法において, 試行回数を増やし, 検定を行うことで, 本当に有意か否かを検討することも後の課題である.

参考文献

[1] Agrawal, S. and Goyal, N.: Analysis of thompson sampling for the multi-armed bandit problem, *Conference on learning theory*, JMLR Workshop and Conference Proceedings, pp. 39–1 (2012).
[2] Auer, P., Cesa-Bianchi, N. and Fischer, P.: Finite-time analysis of the multiarmed bandit problem, *Machine learning*, Vol. 47, No. 2, pp. 235–256 (2002).
[3] Garivier, A. and Cappé, O.: The KL-UCB algorithm for bounded stochastic bandits and beyond, *Proceedings of the 24th annual conference on learning theory*, JMLR Workshop and Conference Proceedings, pp. 359–376 (2011).
[4] Jean Yves Audibert, Rémi Munos, C. S.: Exploration-exploitation trade-off using variance estimates in multi-armed bandits, *Preprint submitted to Theoretical Computer Science* (2018).
[5] Kaufmann, E., Korda, N. and Munos, R.: Thompson sampling: An asymptotically optimal finite-time analysis, *International conference on algorithmic learning theory*, Springer, pp. 199–213 (2012).
[6] Kuleshov, V. and Precup, D.: Algorithms for multi-armed bandit problems, *arXiv preprint arXiv:1402.6028* (2014).
[7] 五ヶ谷純平, 大久保誠也, 若月光夫, 西野哲朗：コンピュータ大貧民におけるレーティング手法について，研究報告ゲーム情報学 (GI)， Vol. 2020, No. 16, pp. 1–8 (2020).
[8] 三石亮，大久保誠也，若月光夫，西野哲朗：コンピュータ大貧民における提出手の影響に関する研究，研究報告ゲーム情報学 (GI)， Vol. 2019, No. 26, pp. 1–6 (2019).
[9] 須藤郁弥，成澤和志， 篠原歩：UEC コンピュータ大貧民大会向けクライアント「snowl」 の開発，第 2 回 UEC コンピュータ大貧民シンポジウム (2010).
[10] 大渡勝己，田中哲朗：方策勾配を用いた教師有り学習によるコンピュータ大貧民の方策関数の学習とモンテカルロシミュレーションへの利用，研究報告ゲーム情報学 (GI)，Vol. 2016, No. 10, pp. 1–8 (2016).
[11] 土橋康希，大久保誠也，若月光夫，西野哲朗：大貧民におけるモンテカルロ法の報酬値に関する研究，研究報告ゲーム情報学 (GI)， Vol. 2019, No. 18, pp. 1–8 (2019).