UCB+ を用いた Big Two AI の研究

万代悠作[†1]　橋本剛[†2]

Big Two とは中華圏を中心に世界中でプレイされているカードゲームであり, 日本のカードゲーム「大貧民」にルールが類似している. しかし Big Two に関する研究は皆無であり, AI の例も少ない. 本研究では大貧民で成功したモンテカルロ法と UCB を組合せた手法を Big Two に適用し, その有用性を示す. さらにより AI を強くするために評価値を UCB に導入した UCB+ とモンテカルロ法を組合せることにより強力な AI を作成し, これらの手法が Big Two において有用であることを示す.

A Study of Big Two AI using UCB+

MANDAI YUSAKU [†1] and HASHIMOTO TSUYOSHI [†2]

Big Two is a popular card game in Chinese culture, which many people enjoy playing with, and the rules of Big Two is very simlar to those of Daihinmin, a Japanese popular card game. However, there are very few studies on this game. In this study, we apply the techniques that the champion AI of Daihimin competition uses to Big Two, and we show the effectivity of these techniques for Big Two. Furthermore, we introduce UCB+ to our AI to improve it. UCB+ is a derivative of UCB, and considers heuristic values. We show that the AI with UCB+ is better than previous one.

1. はじめに

「Big Two」とは多人数不完全情報ゲームに分類され, 世界中でプレイされているカードゲームで, 特に中華圏で人気がある. 後述するルールの特徴により「Chinese Poker」とも呼ばれる. Big Two のルールは日本のカードゲーム「大貧民」のそれによく似ているが, 大貧民と比べてカードの組合せ総数が多く, 強い AI を作成することが難しい. また Big Two は世界的に人気のあるゲームで, オープンソースの AI[1] があるが, 研究はほとんどない. 本研究ではこのゲームを題材に選び, Big Two の 強い AI を作ることを目的とする.

チェスや囲碁などの二人ゲームとは違い, 多人数ゲームでは相手の戦略モデルを適切に選択する必要があり, また相手の行動が必ずしもこちらの損になるわけではないので, 探索が困難になる. また不完全情報ゲームなので不完全な情報をどう扱うかが問題となる. 故に mini-max 法やそれの派生アルゴリズムを適用することは難しい. そこでルールの似ている大貧民の AI「snowl」を参考にした. snowl はコンピュータ AI の大会「電通大コンピュータ大貧民大会 (UECda)」[2] で 2 連覇している強い AI であり, モンテカルロ法 (MC 法) に UCB[3] を用いて手を決定している[4].

本稿では MC 法と UCB を用いて Big Two をプレイする AI のアルゴリズムについて論じる. さらに, MC 法と, 評価値の概念を導入した UCB+ を組合せた手法を提案し, その有効性を示す. 2 章で日本に馴染みのないゲームである Big Two のルールについて解説し, 3 章で我々が参考にした大貧民における関連研究について簡単に説明する. 次に 4 章で UCB, UCB+ について, 5 章で作成した AI と比較するオープンソースの AI について解説する. 6, 7 章で作成した AI の手法について述べ, 8 章で行った対戦実験について述べる.

2. Big Two の概要

この章では Big Two のルール[5] について説明する. また大貧民との相違点についても述べる.

2.1 ルールの概要

Big Two のルールは日本のカードゲーム、大貧民のそれによく似ている. プレイヤーの目的はゲームの最初に配られる手札をすべて場に出すことである. Big Two にも大貧民と同様いくつかのローカルルールが存在するが, 一般的なルールではプレイ人数が 4 人で,

†1 松江工業高等専門学校専攻科 電子情報システム工学専攻
Advanced Engineering Faculty of Electronic and Information Systems, Matsue College of Technology

†2 松江工業高等専門学校 情報工学科
Department of Information Engineering, Matsue College of Technology

ジョーカーを除いた 52 枚のカードでプレイする. 本稿ではカードの数字のこと「ランク」, カードのマークのことを「スート」と定義する. ランク, スートの強さはそれぞれ

$$3 < 4 < 5 < ... < K < A < 2$$

$$\diamondsuit < \clubsuit < \heartsuit < \spadesuit$$

である. カードの強さは最初にランク同士を比較し, ランクの高いものが強い. ランクが同じ場合はスートを比較して強弱を決する.

直前のプレイヤーが場に出したカードの組合せを場札という. カードは後述する組合せで出すことができる. 既に場札がある場合, 場札よりも強いカードの組合せでなければ出すことができず, 出せない場合「パス」しなければならない. また場のカードの組合せと同じ枚数の組合せでないと出すことができない. プレイヤーはカードの組合せを出せる状況であってもパスを宣言することができる. また場にカードを出した後, 他のプレイヤーが全員パスをしたならば場札は流れ, 任意のカードの組合せを出すことができる.

ゲームは 3♢ を持っているプレイヤーから始める. 3♢ を持っているプレイヤーは 3♢ を含む手でなければ出すことができず, パスは許されない. 誰かひとりでも手札がなくなればゲームは終了し, 手札のなくなった一人が勝利者である.

2.2 カードの組合せ

以下に Big Two におけるカードの組合せと組合せの強弱を示す. 以下のいずれかの組合せでなければ出すことができない.

- 1 枚出し (sole)
  1 枚のカードで構成される.
  そのカードの強さで強弱が決まる.
- 2 枚出し (pair)
  同じランクのカード 2 枚で構成される.
  ランクの高い組合せがより強く, ランクが同じ場合は 2 枚のうち強い方のスートを比べる.
- 3 枚出し (triplet)
  同じランクのカード 3 枚で構成される.
  ランクの高い組合せがより強い.
- ポーカーハンド (poker hand)
  5 枚のカードで構成される.
  下にいくほど強い役となる.
  (1) ストレート (straight)
  (2) フラッシュ (flush)
  (3) フルハウス (full house)
  (4) フォー・オブ・ア・カインド (four of a kind)
  (5) ストレート・フラッシュ (straight-flush)

以下にポーカーハンドについて示す.

(1) ストレート
ランクが連番のカードで構成される.
5 枚のカードの中で最高のランクでストレート同士の強弱が決まる. ストレートの判断の時は例外的に 2 が最も弱く, A が最も強い. ランクが同じ場合, 最高のランクのカードのスートで強弱が決まる.

(2) フラッシュ
同スートのカードで構成される.
スートの高い組合せがより強い. スートが同じ場合, 5 枚の内の最高のランクのカードで強弱が決まる.

(3) フルハウス
一組のペアと一組のトリプレットで構成される.
トリプレット部分で強弱が決まる.

(4) フォー・オブ・ア・カインド
4 枚の同ランクカードともう 1 枚の任意のカードで構成される.
ランクの高い組合せがより強い.

(5) ストレート・フラッシュ
ストレートかつフラッシュの場合この役になる.
5 枚のうちで最高のランクのカードで強弱が決まる. ストレートの時と同様に, 2 が最弱で, A が最強である. 同ランクの場合, スートで優劣を決する. 最強のストレート・フラッシュは $T\spadesuit J\spadesuit Q\spadesuit K\spadesuit A\spadesuit$ である.

2.3 ゲームの流れ

以上のルールを踏まえ, ゲームの流れを図 1 を用いて説明する. 図 1 の一番上は 1 枚出しのようすである. 始めに出すプレイヤーが 3♣ を出したとすると, 次のプレイヤーは 3♣ より強いカードを出さなければならない. またスートも強弱に関係するので 2♡ を出したあとに 2♠ を出すことができる. 下二段は 2 枚, 3 枚出しの例である. 1 枚出しの場合と同様に場札よりも強いカードでなければ出すことができない.

ポーカーハンドも同様に, 図 2 において, 始めに出すプレイヤーが 6♢ 7♣ 8♡ 9♠ 10♡ を出したならば, それよりも強い組合せでなければ出せない. 図 2 では 8♣ 9♢ T♠ J♢ Q♡ を出しているが, これは前述の通り同じストレート同士であればランクの高いカードを持つストレートが強いため出すことができている.

2.4 大貧民との違い

以上で説明した一般的なルールと, 後述する UECda[2] のルールの違いを **表 1** に示す.

また他にも違いとして,

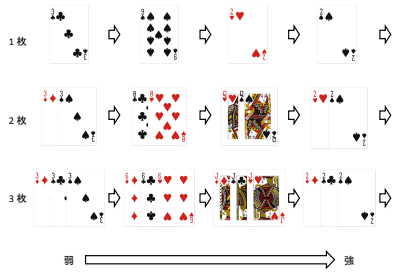

図 1　Big Two の流れ

図 2　Big Two の流れ - ポーカーハンド

表 1　Big Two と UECda のルールの違い

	Big Two	UECda
プレイ人数	4 人	5 人
ポーカーの役	有	無
ジョーカー	無	有
カードの交換	無	有
カードの強さの反転 (革命)	無	有
特殊効果 (8 切り, 3♠ 返し等)	無	有
ロック (場札と同スートを強制)	無	有
階段 (同スート連番 3 枚以上)	無	有

- UECda では一回パスをすると, 場札が流れるまではだせないが, Big Two では自分の手番になれば出せる.
- 大貧民では最後の一人になるまでゲームを続行するが, Big Two では一人上がればゲームが終了する.

などがある.

以上のように, ルール的には単純になっている. しかし **表 2** に示す通り組合せ全体を見ると, 大貧民は全部で 2167 通りの組合せだが, Big Two では 19898 通りの組合せがある. これは表を見れば分かるとおり, Big Two にはポーカーハンドを出すことができるために組合せが膨大になる. そのため局面あたりの平均合法手数も大貧民以上に多くなり, 例えばゲーム開始直後のプレイヤーの平均合法手数は大貧民が約 13.7 手に対し, Big Two は約 27.1 手と約 2 倍になる. この特徴により, 一般に考慮する情報が多くなるので, 強い AI を作成することがより難しくなっている.

表 2　Big Two と大貧民の合法手組合せ数

	Big Two	大貧民
1 枚出し	52	53
2 枚出し	78	130
3 枚出し	52	130
4 枚出し	-	65
ポーカーハンド	19716	-
階段	-	1776
合計	19898	2167

3. 大貧民における関連研究

3.1 UECda

Big Two とルールが類似している大貧民においては 2006 年から電気通信大学主催でコンピュータ大貧民大会 UECda が開催されている[2]. 大会は「自作プログラムの部」と「飛び入り参加の部」があり, どちらもサーバプログラム 1 つにつき 5 つのクライアントプログラムが戦う. 上位プログラムが勝ち抜け, 優勝を決定する. 「自作プログラムの部」では当日までに作ったクライアントプログラムで対戦させ, 「飛び入り参加の部」では当日に運営委員と協力して作成したプログラムで対戦させる. プログラミングの経験のない人でも大貧民のルールを知っているならば大会に参加することができる.

また過去優勝プログラムのソースコードはオープンソースであり, 誰でも閲覧可能である.

3.2 snowl

本研究でプログラムを作成する際に参考にしたプログラム snowl は, 第四回大会で優勝したプログラムを改良した第五回 UECda 優勝プログラムであり, 須藤郁弥氏によって開発された. snowl は場の状況だけでなく, モンテカルロ法と UCB 値を組み合わせたアルゴリズムを元に手を決定しており, ゲーム木の一段目のみ UCT を取り入れていると言える[4].

また snowl は不完全情報の推定に過去の記録から機械学習させた情報を用いており, 精度の高いプレイアウトを実現している[6]. これらの要因によって二大会連続優勝を成し遂げることができたと考えられる.

4. UCB (Upper Confidence Bound)

4.1 UCB

UCB (Upper Confidence Bound) は多腕バンディット問題 (Multi-Armed Bandit Problem) の現実的な解法と

して導かれた[3]. UCB のうちもっとも単純な UCB1 は以下の式で定義される.

$$\mathrm{ucb1}_i = \bar{X}_i + \sqrt{\frac{2\ln n}{n_i}} \tag{1}$$

ここで n_i はノード i を試行した回数, $\bar{X}_i$ は期待値, n は n_i の総和である

4.2 UCB+

UCT 探索では UCB 値の高い手が選択されるが, UCB 値は一般にプレイアウトで得られる報酬の期待値が大きなウェイトを占める. しかし報酬の期待値だけではプレイアウト数を大きくしなければ誤差が大きく, 良い結果が得られるまで時間がかかる. そこでその局面の評価値を UCB 値に組み合わせ, 局面評価値上良いと思われる局面の探索を早く行おうとするアイディアが UCB+である. 局面 P で i 番目の子局面が局面評価値 E_i を持ち, n_i 回のプレイアウトを行った時点での報酬の期待値を $\bar{X}_i$, 全部で n 回のプレイアウトを行なっているとすると, $\mathrm{ucb}_\mathrm{i}^+$ 値は次のようになる.

$$\mathrm{ucb}^+{}_i = \bar{X}_i + C\sqrt{\frac{2\ln n + E_i}{n_i}} \tag{2}$$

UCB+は UCT 探索に用いられ, UCT+ としてオセロにおいて性能向上を示し, その有効性が示されている[7].

5. 比較対象の AI

5.1 概　　要

Big Two のプログラムはスマートフォン向けのアプリケーションなどがいくつか存在するが, オープンソースではない. 今回は Bernard Yap 氏作成の Big Two プログラム「coffee[1]」中の AI (以下 coffee AI) を比較対象とした. このプログラムはオープンソースであり, 誰でも参照できる.

5.2 評 価 関 数

AI は手の選択に評価関数を用いており, 手を選択するときにすべての合法手を評価関数によって評価し, もっとも良い手を選択する. coffee AI の評価関数では以下の項目を考慮している.

- 評価する手を除いた後の手札 (残り手札) を全て出すために必要な最小手番数
- 残り手札の枚数
- 残り手札を構成するカードの強さ
- 強いカードをどのくらい使っているか
- 自分が連続してパスした回数
- この次の自分の手番の際にカードが出せるか否か

具体的な coffee AI の手の評価を以下の計算式によって示す.

$$E = E_{sp} + \frac{E_{sz}}{C} + \frac{1000 - E_{sum}}{C^2} + E_p \tag{3}$$

$$\tag{4}$$

ここで, E_{sp} は残り手札をすべて出すために必要な最小手番数, E_{sz} 残り手札の枚数, E_{sum} は残り手札中のカードの強さの総和, C は定数で, $C = 1000$ である. E_p は強いカードをどのくらい使っているか, パスした回数などを考慮した評価値である. カードの強さは 3♢ が 0 , 3♣ が 1 で, 以下カードがひとつ強くなるごとに 1 増える. したがって 2♠ が 51 となる.

coffee AI は E が最も低い手を出すようにしている.

6. MC 法と UCB を用いた AI

はじめに作成した AI は MC 法と UCB 値を組み合わせたアルゴリズムを用いる UCB AI である. 前述のとおり大貧民大会ではこのアルゴリズムを用いた AI snowl が二大会連続で優勝している.

今回は snowl と同じく UCB1-tuned 値を用いた. UCB1-tuned 値は UCB1 値にノードの報酬の分散を取り入れて精度が上がっている. ノード i について n_i 回プレイアウトを行った時点でのそのノードの報酬の期待値を $\bar{X}_i$, 分散を V_i とし, 全部で n 回プレイアウトを行なっているとするとそのノードの UCB1-tuned 値は次のようになる.

$$\text{ucb1-tuned}_i = \bar{X}_i + C\sqrt{\frac{\min(V, 0.25)\ln n}{n_i}} \tag{5}$$

ただし

$$V_i = \bar{X_i^2} - (\bar{X}_i)^2 + \sqrt{\frac{2\ln n}{n_i}} \tag{6}$$

とし, C は定数である.

この値を用いてプレイアウトするノードを決定する. 始めにノードごとにこの値を計算し, 最も値の大きいノードをプレイアウトする. そして報酬を観測し, 値を更新してもう一度最大のノードを選択していく. これを一定回数繰り返し, 最も勝率の高いノードを手として選ぶ. これにより, 見込みのない手にはあまりプレイアウト回数が割り当てられず, 図 3 のように見込みの高い手ほどプレイアウトの回数が増えるようになっている.

7. MC 法と UCB+ を用いた AI

7.1 UCB+

性能を向上させるため, プレイアウトノードを決定

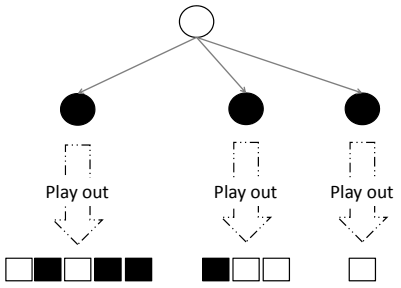

図 3 UCB と モンテカルロ法の組合せ

する UCB を改良し, UCB+ を導入した UCB+ AI を作成した. UCB+ は評価値を取り入れることによってより早い段階で良いノードを選別できるように改良された値である. 本研究では以下の式で UCB+ 値を求める.

$$\mathrm{ucb}+_i = \bar{X}_i + C\sqrt{\frac{\min(V, 0.25)\ln n + C_E E_i}{n_i}} \quad (7)$$

手の評価値をプレイアウト回数で除することにより, プレイアウト回数が増えれば増えるほど局面評価値は小さくなり, 勝率が重視されるようになる. これによりプレイアウト回数が少ない初めの試行では評価値の高い, すなわち良いと思われるノードに, 多い後の試行では勝率の高いノードに多くのプレイアウトを割くことができる. オセロにおいては UCB1 を改良して評価値を組み込んでいたが, 今回は UCB1-tuned に評価値を組み込んで前述の AI との比較をできるようにした.

7.2 評 価 値

評価値は coffee AI のものを基に, 新たに以下の項目を評価に加えた.

- 相手手札の残り枚数
- その手がより多くの枚数を使うカードの組合せを崩していないかどうか

これは coffee AI の傾向としてゲーム序盤で他のプレイヤーが勝負を賭けにきているときでもパスする傾向が強かったために, またより早く上がるために重要だと考えられるポーカーハンドなどの大手物を崩してしまわないように導入した. 局面評価値 E_i は 0.0 から 1.0 までの値を取るよう調整した.

8. 実 験

UCB AI, UCB+ AI と coffee AI を対戦させた.

8.1 実 験 環 境

実験は以下の環境で行った. また AI の実装には C++ 言語を用いた.

- OS: Windows 7 Enterprise 64bit
- CPU: AMD AthlonII X4 620 Processor 2.60GHz
- メモリ: 4GB
- 開発環境: Microsoft Visual Studio 2010 Ultimate

また対戦は 3 のように 2 つの AI を対戦させた. 各対戦は 500 回行った.

表 3 対戦実験の組合せ

プレイヤー A	プレイヤー B	プレイヤー B	プレイヤー B
プレイヤー A	プレイヤー A	プレイヤー B	プレイヤー B
プレイヤー A	プレイヤー A	プレイヤー A	プレイヤー B

8.2 UCB AI

はじめに UCB AI と coffee AI を対戦させた. UCB AI は式 (5) 中の C の値を変化させて対戦させた. またプレイアウト回数は 2000 回に固定して対戦をさせた. 表に結果を示す.

表 4 UCB AI と coffee AI との対戦

C	勝率
0.8	0.5953
1.0	0.5987
1.2	0.5840
1.4	0.5880
1.6	0.5993
1.8	0.5874

$C = 1.6$ のときに勝率が最大となった.

8.3 UCB+ AI

UCB AI を改良した UCB+ AI と, coffee AI を対戦させた. 式 (7) 中の定数は, $C = 1.6$ とし, C_E を変化させて対戦させた. プレイアウト回数は同様に 2000 回とした. 表に結果を示す.

表 5 UCB+ AI と coffee AI との対戦

C_E	勝率
0.3	0.632
0.6	0.611
1.0	0.597
1.3	0.570
1.6	0.568
1.9	0.540

また UCB+ AI と UCB AI を対戦させた結果も以下の表に示す. $C = 1.6, C_E = 0.3$ とした.

評価値を考慮した UCB+ AI のほうが UCB AI よりもよい結果を残していることが分かる.

表 6　UCB AI と UCB+ AI との対戦

	UCB AI	UCB+ AI
勝率	0.452	0.548

9. 考　　察

9.1 UCB AI

8.2 章では大貧民で成功した手法を用いた AI を検証したが, 大貧民だけでなく, Big Two でも有効な手法であることを示せた. ルール自体が類似しているため, Big Two 以外の大貧民のようなゲームでも有効だと考えられる. 現在の snowl は機械学習を用いてプレイアウトや相手手札の推定をより現実に近いものにしているが, 今回はランダムにプレイアウトや相手手札を推定した. ゲームに関する知識がなくともある程度の強さをもつ AI を作成できた.

9.2 UCB+ AI

8.3 章では評価値を考慮した UCB+ を導入し, 性能を上げることができた. 今回の評価値は相手の状況をあまり考慮していないが, それでも性能の向上が見られた. UCB+ のほうが良い成績を残せた原因としては, Big Two は局面あたりの合法手の数が多く, UCB と MC 法だけでは良い手に収束することが難しいが, 評価値がある UCB+ と MC 法では早い段階で良い手にプレイアウトを割り当てることができるため, ということが予想される. MC 法で多人数不完全情報ゲームをランダムにプレイアウトしても, 相手の手札や行動が不確定であることもあって十分回数プレイアウトをしないと信頼できる結果は返ってこない. その点 UCB+ を使うことでもとより見込みのある手を優先的にプレイアウトするので, よりよい成績を残せたのではないかと考える.

UCB+ AI において $C_E = 0.3$ のときに勝率が最大になり, 以後増えるごとに勝率は下降していった. オセロで実証した際には $C_E = 1.0$ であったことを考えると, ゲームに依った適切な値でないとよい結果を与えないことがわかる.

10. 終わりに

本研究で大貧民で成功したモンテカルロ法と UCB を組合せたアルゴリズムが Big Two でも有効であることがわかった. また UCB 値を UCB+ 値に変更することでより強い AI を作成することができた.

今後の展望としてはアルゴリズムの改良が挙げられる. UCB と MC 法ではなく, UCT アルゴリズムを Big Two に適用することがまず考えられる. UCT を skat[8] や麻雀[9] に適用した例 などがあり, それらの手法を Big Two などのカードゲームにも適用できるかどうかを調査する. また囲碁[10] や大貧民[6] で成功したように機械学習を用いてプレイアウト中の行動や相手手札の推定などをより現実に近いものとすることがあげられる. 不完全情報ゲームにおいては典型的なアプローチであるが, それらを用いて評価値の精度を高め, UCB+ と MC 法を用いたアルゴリズムをより強力にする, などのアプローチも考えられる.

参　考　文　献

1) SourceForge.net: bigtwo - Project Web Hosting - Open Source Software, `http://bigtwo.sourceforge.net/`.
2) UEC UEC コンピュータ 大貧民大会, `http://uecda.nishino-lab.jp/`.
3) 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).
4) 須藤郁弥, 篠原歩 : モンテカルロ法を用いたコンピュータ大貧民の思考ルーチン設計, 第一回 UEC コンピュータ大貧民シンポジウム (2009).
5) Rules of Card Games: Big Two, `http://www.pagat.com/climbing/bigtwo.html`.
6) 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, 第二回 UEC コンピュータ大貧民シンポジウム (2010).
7) 前原彰太, 橋本剛, 小林康幸 : 局面評価関数を使う新たな UCT 探索法の提案とオセロによる評価, 情報処理学会研究報告. GI, [ゲーム情報学], Vol. 2010, No.5, pp. 1–5 (2010-06-18).
8) Schäfer, J.: The UCT algorithm applied to games with imperfect information, *Diploma, Otto-Von-Guericke Univ. Magdeburg, Magdeburg, Germany* (2008).
9) 三木理斗, 三輪誠, 近山隆 : UCT 探索による不完全情報下の行動決定, 第 14 回ゲームプログラミングワークショップ, pp. 43–50 (2008).
10) Coulom, R.: Computing Elo Ratings of Move Patterns in the Game of Go, in Herik, van den H. J., Winands, M., Uiterwijk, J. and Schadd, M. eds., *Computer Games Workshop*, Amsterdam, Pays-Bas (2007).