愛知工業大学研究報告
第 58 号 令和 5 年

＜査読なし論文＞

拡張モンテカルロ木探索によるブラックジャック戦略

Blackjack Strategy with Extended Monte-Carlo Tree Search

伊藤 雅 †
ITOH Masaru†

Abstract: This paper focuses on card games, especially Blackjack, which are representative of imperfect information games. AlphaZero, which developed AlphaGo, conquered Go, the most difficult perfect information game. It owes much to Monte-Carlo tree search. The idea of Monte-Carlo tree search, one of the best-first search methods, originates from the UCB algorithm for the multi-armed bandit problem. Using the extended Monte-Carlo tree search, the search tree is improved so as to derive the best action of Blackjack, which is one of the imperfect information games, and the algorithm is constructed. One playout performed at each node of the game tree corresponds to a player winning or losing in that round. The purpose is to realize a Blackjack strategy that can compete with the basic strategy that is effective in Blackjack by searching an extended Monte-Carlo tree. In numerical experiments, the proposed method is evaluated by using the payout ratio.

1. はじめに

本論文は不完全情報ゲーム [1, 2, 3] の代表格であるトランプゲーム、特にブラックジャックをその研究対象とする。完全情報ゲームの最高峰であった囲碁をアルファゼロ [4] で攻略できたのはモンテカルロ木探索 (Monte-Carlo Tree Search: MCTS)[5] に負うところが大きい。アルファゼロは 2016 年に発表された教師付き深層学習を用いたアルファ碁 [6] やその翌年に発表された強化学習のみのアルファ碁ゼロ [7] を鋭意発展させたゲーム戦略 AI である。2018 年に発表されたこのゲーム戦略 AI の中心的役割を担っているのがモンテカルロ木探索である。もうひとつの核心が深層学習 (Deep Learning)[8] である。最良優先探索手法のひとつであるモンテカルロ木探索のアイデアは多腕バンディット問題（Multi-armed bandit problem）[9] に対する UCB アルゴリズム [10] に端を発している。

最良優先探索で不完全情報ゲームを攻略できるようにブラックジャックのゲーム木を改良・拡張し、そのアルゴリズムを構築する。目的はブラックジャックで有効とされているベーシックストラテジー [11] やカードカウンティング [12] に対抗し得る方法を拡張モンテカルロ木探索で実現することである。

数値実験における提案手法の評価では、ペイアウト率（payout ratio）を使用する。ペイアウト率とは、賭け金総額に対する払戻金総額の比をパーセント表示した指標である。還元率、払戻率、期待値ともいう。ブラックジャックではベーシックストラテジーやカードカウンティングといった既存の手法を駆使して上手くプレイすれば、プレイヤーのペイアウト率が 100%を超す場合もあることが統計的あるいは実践的に証明されている [11]。

† 愛知工業大学 情報科学部 情報科学科（豊田市）

2. ブラックジャックゲーム

ブラックジャックゲームは 2 枚以上のカードの合計が ‘21’ に近い方が「勝ち」、‘21’ を超えれば「負け」というゲームである。ひとりまたは最大 4 人のプレイヤーとディーラーが対戦する状況を想定する。使用するカードは ♠、♣、♡、♢ の 4 スーツとランクが $A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K$ の 13 枚からなる計 52 枚を 1 デッキとする最大 6 デッキである。A だけは ‘1’ または ‘11’ のどちらにもカウントできる。J, Q, K の 3 つはすべて ‘10’ とカウントする。プレイヤーが 2 枚で 21 勝ちのブラックジャックのとき、賭け金の 1.5 倍が払い戻される。

ゲームの進行を図 1 を使って説明する。

デッキに収納されている残りカード枚数が事前設定枚数以下になるまで連続プレイする。設定枚数以下になると 1 セットが終了する。すべてのプレイヤーがディーラーと勝負し終わるまでが 1 ラウンドである。1 ラウンドの最後に

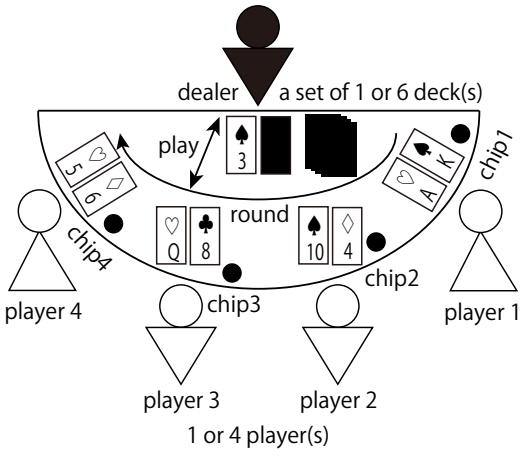

図 1 ブラックジャックゲームの流れ

賭け金が精算され、そのラウンドでの全プレイヤーの勝敗と払戻金が確定する。ラウンドの途中でプレイは終了しない。事前設定したセット回数終了時またはラウンド回数終了時にプレイヤーの勝敗数とペイアウト率を算出する。ペイアウト率は賭け金の総額と払戻金の総額の比から式 (1) によって計算される。

$$\text{ペイアウト率} = \frac{\text{払戻金総額}}{\text{賭け金総額}}. \tag{1}$$

まず、プレイヤー全員が賭け金となるチップをテーブルに置く。ディーラーは時計回りにプレイヤー全員とディーラー自身にカードを 2 枚ずつ配る。1 枚目をアップカード（表向きカード）で配り、2 枚目はプレイヤーにはアップカード、ディーラー自身にはダウンカード（伏せたカード）で配る。デーラーのダウンカードをホールカード（hole card）ともいう。この状態からラウンドが開始する。

次に、ディーラーは第 1 プレイヤーから順にカードをアップカードで配り、プレイヤーの手（ハンド）を順次確定していく。第 1 プレイヤーがスタンドを宣言すれば、第 1 プレイヤーの手が確定する。ヒットを宣言すれば、ディーラーは新たなカード 1 枚をデッキ（deck）から抜き取りアップカードでプレイヤーに配る。ヒットは連続して宣言可能である。

スタンドやヒット以外にもスプリット（sprit: 同位札分割）、ダブルダウン（double-down: 倍賭けで 1 ヒットのみ）、サレンダー（surrender: 降参して賭け金の半分を没収）といった行動もプレイヤーは選択できる。カードの合計点が 21 を超えた時点でプレイヤーの「敗北」が確定し、賭け金は没収される。これをバスト（Bust）という。プレイヤーがスタンドまたはバストすれば、ディーラーは次のプレイヤーに対象を移す。図 1 は第 3 プレイヤーが自身のハンドを確定するためにプレイしている状況を示している。プレイヤー全員のハンドが確定すれば、最後にディーラー自身がハンドを確定する。

ディーラーが勝負するのはバストしなかった残りのプレイヤーだけである。プレイヤーとの違いは、ディーラーは合計点が 17 未満で必ずヒット、17 以上で必ずスタンドという制約がある。つまり、ディーラーにはヒットやスタンドといった行動の選択肢が一切ない。一方のプレイヤーはどの時点でもスタンドを宣言できる。合計点が 17 以上という制約もない。同点ならば引き分けで賭け金は戻る。

ブラックジャックのゲーム木を図 2 に示す。ここで、スプリットだけは特別扱いとする。通常のスプリットでは 2 枚の初期ハンドを 2 つに分け、別々にプレイする。スプリットしたプレイヤーからすれば、連続して 2 回プレイすることができる。スプリットした後、ディーラから新たにカード 2 枚が配られ、2 組の初期ハンドを得る。ゲーム木では第 1 ハンドが確定した後で第 2 ハンドのプレイがスタートする、と想定する。しかも、簡単化のためスプリットでは選択できる行動を {stand, hit} の二択に限定する。

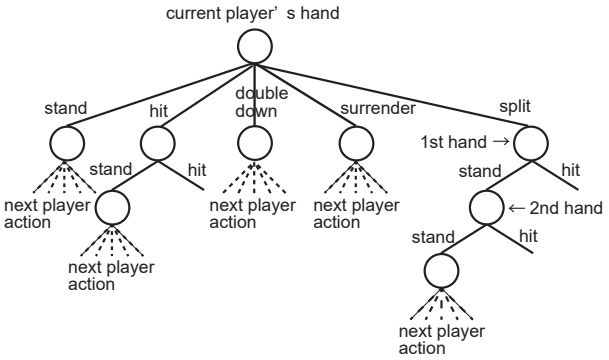

図 2 ブラックジャックのゲーム木

3. ベーシックストラテジー

ブラックジャックの研究は古く、1960 年代にエドワード・O・ソープ [13] が書籍「Beat the Dealer」の中でブラックジャックの基本戦略について述べている。プレイヤーのもつ 2 枚のハンドとディーラーのアップカード 1 枚から統計的手法を駆使してプレイヤーの行動を一意に決定する。これがベーシックストラテジーである。

ベーシックストラテジーでは 3 つの表を利用する。ハードハンド（表 1)、ソフトハンド（表 2)、ペアハンド（表 3) に対応した 3 つの表である。A が 1 または 11 のどちらか一意にしか解釈できないときに利用するのがハードハンド表である。A が 1 または 11 のどちらにも解釈できるときに利用するのがソフトハンド表である。例えば、表 2 にある A・5 は、A を 1 と考えれば 6 であり、A を 11 と考えれば 16 である。6 or 16 と書けば、これが Soft Total である。3 つ目のペアハンド表は初期の 2 枚のカードが同位札のときに利用する。ペアハンド表にはスプリットすべきか否かの指標 SP も含まれる。SP の場合はスプリットを選択する。3 つの表を埋めている 5 つの記号は次の通りである。S: スタンド（Stand)、H: ヒット（Hit)、D: ダブルダウン（Double down)、SP: スプリット（SPrit)、SR: サレンダー（SuRrender)。デーラーのアップカード 10 はカード 10/J/Q/K の代表値である。これら 4 つのカードはすべてランク 10 カードとして同様に扱われる。

3 つの表の使い方を説明する。はじめにペアハンド表を使い、該当がなければ、次にソフトハンド表を使う。それも該当しなければ、最後にハードハンド表を使う。これでプレイヤーの行動を一意に決定できる。これ以降は基本的にハードハンド表に従う。再スプリットできないと仮定すれば、スプリットの場合、自身のプレイが連続 2 回チャレンジできると考えればよい。

4. カードカウンティング

ベーシックストラテジーでは、デッキ内に収まっているカードについては一切考慮しない。表向きで開示されているディーラーのアップカードと自分のハンドにあるアップカードのみに依存して統計に基づいた最適行動をプレイ

表 1 ベーシックストラテジー（ハードハンド）

		Up card of the dealer									
	Hard Total	2	3	4	5	6	7	8	9	10	A
Hard hand	5 ∼ 8	H	H	H	H	H	H	H	H	H	H
	9	H	D	D	D	D	H	H	H	H	H
	10	D	D	D	D	D	D	D	D	H	H
	11	D	D	D	D	D	D	D	D	D	H
	12	H	H	S	S	S	H	H	H	H	H
	13	S	S	S	S	S	H	H	H	H	H
	14	S	S	S	S	S	H	H	H	H	H
	15	S	S	S	S	S	H	H	H	SR	H
	16	S	S	S	S	S	H	H	H	SR	SR
	17 ∼ 21	S	S	S	S	S	S	S	S	S	S

S: Stand, H: Hit, D: Double down, SR: Surrender,
Up card 10: {10, J, Q, K}

表 2 ベーシックストラテジー（ソフトハンド）

		Up card of the dealer									
		2	3	4	5	6	7	8	9	10	A
Soft hand	A · 2	H	H	H	D	D	H	H	H	H	H
	A · 3	H	H	H	D	D	H	H	H	H	H
	A · 4	H	H	D	D	D	H	H	H	H	H
	A · 5	H	H	D	D	D	H	H	H	H	H
	A · 6	H	D	D	D	D	H	H	H	H	H
	A · 7	H	D	D	D	D	S	S	H	H	S
	A · 8	S	S	S	S	S	S	S	S	S	S

S: Stand, H: Hit, D: Double down,
Up card 10: {10, J, Q, K}

表 3 ベーシックストラテジー（ペアハンド）

		Up card of the dealer									
		2	3	4	5	6	7	8	9	10	A
Pair hand	2 · 2	H	H	SP	SP	SP	SP	H	H	H	H
	3 · 3	H	H	SP	SP	SP	SP	H	H	H	H
	4 · 4	H	H	H	H	H	H	H	H	H	H
	5 · 5	D	D	D	D	D	D	D	D	H	H
	6 · 6	H	SP	SP	SP	SP	H	H	H	H	H
	7 · 7	SP	SP	SP	SP	SP	SP	H	H	H	H
	8 · 8	SP	SP	SP	SP	SP	SP	SP	SP	SP	SP
	9 · 9	SP	SP	SP	SP	SP	S	SP	SP	S	S
	10 · 10	S	S	S	S	S	S	S	S	S	S
	A · A	SP	SP	SP	SP	SP	SP	SP	SP	SP	SP

S: Stand, H: Hit, D: Double down, SP: Sprit,
Up card 10: {10, J, Q, K}

ヤーは選択する。

ブラックジャックではデッキ内の残りカードがプレイヤーに有利なときには大きく賭け、不利なときには小さく賭けた方がよい。残りカードにランク 10 のカードが多ければ多いほどプレイヤーにとっては有利となり、逆に、低ランクカード (2〜7) が多ければ多いほどプレイヤーにとっては不利になる。そこで、ディーラーが配るアップカードを目視で確認して、低ランクカード (2〜7) には正値を与え、高ランクカード (10, J, Q, K, A) には負値を与えて、その合計から賭け金を適宜変動させる。そうすればプレイヤーの獲得金額はより多くなり、ペイアウト率は向上する。これがカードカウンティングという戦略である。どのタイミングで賭け金を多くし、どのタイミングで賭け金を少なくするかを決定しなければならない。

カードカウンティングを実行するには、カウント値（count value）を事前に与えておく必要がある。最も簡単なのが $\{\pm 1, 0\}$ を使った表 4 である。$\{-3, 0, +1, +2, +3\}$ を使ってシミュレーション向きに修正したのが表 5 である。このカウント値をアップカードが出現する度に加算してランニングカウント（Running Count: RC と略記）を式 (2) で更新する。

$$RC \leftarrow RC + count_i, \quad i = 1, 2, \cdots N. \tag{2}$$

ここで、$i = 1$ はディーラーがデッキセットを交換後に配る 1 枚目を意味し、$i = N$ はデッキセットを交換する直前のラウンドでディーラーが最後に配った N 枚目を意味する。$count_i$ はディーラーが第 i 番目に配ったアップカードのカードランク（card rank）に対応するカウント値である。当然ながら、RC の計算はゲームの進行に合わせて逐次更新することになる。

RC の初期値が IRC（Initial Running Count）である。デッキセットが交換されれば、言い換えれば、1 セットが終了すれば、$RC = IRC$ にリセットされる。

ランニングカウント RC の初期値 IRC は文献 [12] を参考に式 (3) のように設定する。

$$IRC = 4 - (4 \times decks), \quad decks\text{: 使用デッキ数.} \tag{3}$$

1 デッキのとき初期値 0 から RC をカウントし始め、6 デッキでは RC を −20 からスタートする。式 (3) の導出根拠は次のように考えればよい。1 デッキ 52 枚のすべてのカウント値の総合計は表 4 を使えば、+4 となる。1 デッキの IRC を恣意的に 0 になるようにするには、この定数 4 を使って $IRC = 0 = 4 - (4 \times deck)$, $deck = 1$ とすればよい。これが式 (3) の導出根拠である。

RC を使って賭け金 5/10/50/150/250/300 を適宜変動させるには、例えば表 6 のように設定する。

表 6 を 1 デッキの場合で説明すれば、RC 値が 0 以下ならば、賭け金を最小の 5 にする。理由は、負値となる高ランクカード (10, J, Q, K, A) がすでに多数出現しているので、次ラウンドでプレイヤーに配られる 2 枚のハンド合計が 21 に近いとは考え難い。何枚もヒットして自身がバストする可能性が高いので賭け金を低く抑えるのである。逆に、RC 値が 5 以上ならば、既出カードは低ランクカード (2〜7) が多く、デッキ内にはまだ多くのランク 10 カードが残っているので、プレイヤーには有利と判断できる。

このようにカードカウンティングは賭け金を決定する手法であって、プレイヤーの行動（ヒットやスタンドなど）を決定するものではない。行動の選択にはベーシックスト

表 4 ランクに基づくカウント値 ±1

card rank	2	3	4	5	6	7	8	9	10	J	Q	K	A
count value	+1	+1	+1	+1	+1	+1	0	0	−1	−1	−1	−1	−1

表 5 ランクに基づくカウント値 ±3

card rank	2	3	4	5	6	7	8	9	10	J	Q	K	A
count value	+2	+2	+3	+3	+1	+1	0	0	−3	−3	−3	−3	−3

表 6 ランニングカウント RC による賭け金の変動

RC for 1 deck	0 or less	1	2	3	4	5 or more
betting chip values	5	10	50	150	250	300
RC for 6 decks	−20 or less	−19 ∼ −5	−4	−3	−2	−1 or more
betting chip values	5	10	50	150	250	300

表 7 勝敗に基づくプレイアウトの報酬と賭け金倍率

win or lose	Reward of playout	return rate
win (double down)	+8	2×
win (blackjack)	+7	1.5×
win (normal)	+6	1×
draw	+4	0
lose (surrender)	+3	−0.5×
lose (normal)	+2	−1×
lose (double down)	0	−2×

ラテジーを利用するしかない。これがベーシックストラテジーとカードカウンティングの併用である。

5. 提案法の概要

多腕バンディット問題 [9] に対する決定論的方策の一つに UCB アルゴリズム [10] がある。UCB とは Upper Confidence Bound の略である。単純なモンテカルロ法では深さ 1 の先読みしかできない。そこで UCB アルゴリズムとモンテカルロ法をゲーム木探索に組み込む。UCB アルゴリズムを組み込んだ木探索が UCT（UCB applied to Trees）アルゴリズム [14] であり、これを一般にモンテカルロ木探索と呼ぶ。UCB アルゴリズムを用いて適切なノードを選択しプレイアウトを実行する。プレイアウトの報酬は勝敗 $\{+1, 0\}$ でもよいし、賭け金から得られる利益または損益でもよい。勝敗ならば、平均報酬はそのまま勝率と捉えることができる。囲碁では勝率が圧倒的に有利であることが知られている [5]。提案法では表 7 に示す勝敗に基づくプレイアウトの報酬（0 ∼ +8）についても提案する。

UCB アルゴリズムで使われるノード v の UCB_v の定義を式 (4) に示す。ここで、$\overline{X}_v$ はノード v の平均報酬、n_v はノード v 以降のプレイアウト回数、n は v の親ノードでのプレイアウト総数である。式 (4) の右辺第 1 項はノード v の平均報酬値、第 2 項はバイアス値と捉えればよい。

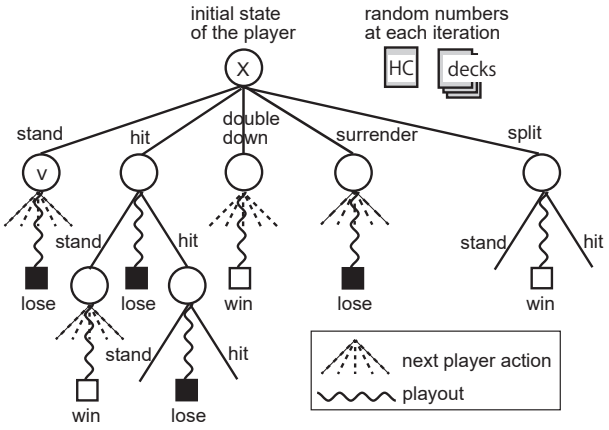

図 3 拡張モンテカルロ木探索の探索木

$$UCB_v = \overline{X}_v + C\sqrt{\frac{2\ln n}{n_v}}, \quad C\text{: constant.} \tag{4}$$

図 3 は現在のハンドで最善行動を求めようとしている。簡単に説明する。まず、根ノード（root node）に手持ちハンド X が登録される。アークがプレイヤーの行動であり、ノードがその行動を取った後のハンドを示す。根ノード登録時には内部ノードは存在せず、根ノード直下にすべての選択可能な複数行動が葉ノードとして追加される。葉ノードが追加されると根ノードに登録されたハンドからアークに沿った行動を選択した後、プレイアウトが 1 回だけ実行される。スタンドしたノード v から真下に降りる波線がプレイアウトである。プレイアウトとは、あるハンドからの行動選択以降、そのラウンドが終了するまで自分を含む全プレイヤーが適当に行動を選択して勝敗が喫するまでの 1 回のシミュレーションのことをいう。

プレイアウト中のプレイヤーの行動はベーシックストラテジーに従う。ここにランダム性はない。後続プレイヤーの行動もベーシックストラテジーで決定する。この部分が図 3 の探索木に描かれているプレイアウト（波線部分）の中身である。プレイアウトの最後でディーラーのハンドが

17 以上になればその仮想ラウンドが終了し、勝敗が喫し、賭け金が精算される。このときそのプレイアウトの報酬が確定する。プレイアウト報酬は必ずしも賭け金の精算額である必要はない。勝敗による報酬 {+1, 0} でもよい。

プレイアウトによるノードの評価が済めば、その結果を根ノードまで順次伝播する。これが木の更新である。更新が終了したら、根ノードから再び式 (4) の UCB 値の大きい子ノードに降りることになる。葉ノードに到達すれば、その葉ノードを初期ハンドとしてまたプレイアウトを 1 回実行し、その結果を根ノードまで返す。

葉ノードに到達する回数が一定回数以上になれば、葉ノードを展開して子ノードを追加し、木が 1 段深くなる。この事前に決定しておく一定回数のことをノード展開閾値（node expansion threshold）という。葉ノード生成時には新たなノードで必ずプレイアウトが 1 回だけ実行される。葉ノードが展開されると、このノードは内部ノードとなる。通常のモンテカルロ木では内部ノードではプレイアウトは実行されない。プレイアウトの試行は葉ノードに限定される。しかし、ブラックジャックでは、ヒットの行動を選択した場合、追加されるカードのランクによってはバストしてしまい、その時点で敗北が確定する。そこで、バストしたときにはその内部ノードでもプレイアウトを 1 回実行し、プレイアウト報酬を決定して、その報酬を根ノードまで遡及させる。これが提案する拡張モンテカルロ木探索の最大の特徴である。

プレイアウトさせた葉ノードまたは内部ノードから根ノードに至る経路上にあるすべての内部ノードで UCB 値が逐次更新される。これを一定回数繰り返す。プレイアウト回数が事前に定めたプレイアウト上限回数に達したとき、木探索が終了する。木探索が終了した時点で根ノード直下の子ノード到達回数が最大の行動をプレイヤーは採択する。それがスタンドならば、次プレイヤーに手番が進む。ヒットならば、新たなカード 1 枚がディーラから配られ、実ハンドが更新される。この場合、次の行動の決定には再び拡張モンテカルロ木探索を実行することになる。

拡張モンテカルロ木探索のランダム性は、ディーラーのホールカード（図 3 の記号 HC）とデッキに収納されている残りカードの並び（図 3 の記号 decks）の 2 つである。根ノードから木を降り出す直前に毎回これら 2 つを乱数で決定する。囲碁のプレイアウトようにランダムに盤面に石を置いて終局させて勝敗を決めるのではなく、ブラックジャックのプレイアウトでは、ホールカードとデッキの残りカードの並びをランダムに設定して、そのラウンドを終了させて勝敗と賭け金を精算する。

不完全情報ゲームであるブラックジャックに対し、提案する拡張モンテカルロ木探索を用いたブラックジャック戦略のアルゴリズムをまとめると次のようになる。

提案法アルゴリズム

Step0: 根ノードに現在のプレイヤーの実ハンドを登録し、ディーラーのホールカード（図 3 の HC）とデッキの残りカードの並び（decks）を乱数で初期化する。さらに、深さ 1 の子ノードを生成する。生成と同時に各葉ノードでプレイアウトを 1 回だけ実行する。これで根ノード直下に配置された葉ノードの UCB 値が初期設定される。

Step1: 根ノードから有望な子ノード、具体的には UCB 値が最大の子ノードを選択する。選択候補は {stand/hit/double down/surrender/sprit} の 5 つである。再スプリットを考慮しないため、スプリットはプレイヤーの実ハンドが同位札の場合に限られる。

Step2: 子ノードに遷移したらアークにある選択行動を実行する。ここでもしバストしたら、その内部ノードでプレイアウトを 1 回実行して Step5 に進む。

Step3: バストしなければ、再び UCB 値最大の子ノードを選択して、一段だけ木を降りる。葉ノードに到達するまで Step2∼Step3 を繰り返す。

Step4: 辿り着いた葉ノードのプレイアウト回数がノード展開閾値を超えれば、その葉ノードを展開して探索木を一段深くする。このとき葉ノードは内部ノードになる。追加した子ノードのすべてで 1 回だけプレイアウトを実行して、全子ノードの UCB 値を決定する。もしノード展開閾値を超えていなければ、その葉ノードでプレイアウトを 1 回だけ実行する。

Step5: そのノードでのプレイアウトの結果から UCB 値を更新する。併せてそのノードでのプレイアウト回数も更新する。それらの値（UCB 値とプレイアウト回数）を根ノードまで順に遡及させる。結果的に根ノードまでの経路上にあるすべての内部ノードの UCB 値とプレイアウト回数が更新されることになる。

Step6: プレイアウト回数が事前設定したプレイアウト上限数に達していれば Step7 へ、さもなければ、ディーラーのホールカード（HC）とデッキの残りカードの並び（decks）を乱数で再初期化して Step1 に戻る。

Step7: 根ノード直下の子ノードがもつプレイアウト回数が最大のノードを選択する。そして、そのノードへの行動を最適行動として採択する。ここでアルゴリズムが終了する。hit/double down/split を採択すれば、ディーラーから新たなカードが実際に配られる。

Step0, Step2, Step4 でプレイアウトを実行している。プレイアウトでは、プレイヤーのハンドにペアハンド表 → ソフトハンド表 → ハードハンド表を順に適用してベーシックストラテジーを基準に該当する行動を一意に選択する。ハンドが 3 枚以上ならば、ハードハンド表のみに依存する。後続プレイヤーの行動（図 3 にある next player action）もすべてベーシックストラテジーで決定する。

重要なのは、そのラウンドが終了するまでがプレイアウトである、という点である。正確には、ディーラーのハンドが確定して全プレイヤーの賭け金が精算されるまでが 1 プレイアウトである。ここにランダム性は一切ない。提案アルゴリズムの中でランダム性を持たせているのはディーラーのホールカードとデッキの残りカードの並びを乱数で

初期化する Step0 と Step6 だけである。Step0 はアルゴリズムをスタートさせるための初期化処理であり、Step6 は Step1 に戻るための再初期化処理である。

6. 数値実験

6・1 実験の前提条件と仮定

提案した拡張モンテカルロ木探索（Extended Monte-Carlo Tree Search: eMCTS）を用いたブラックジャック戦略のアルゴリズムを 2 種類用意した。ひとつはプレイアウトの報酬を勝敗（win-lose）で与える eMCTS-WinLose であり、もうひとつはプレイアウト報酬を表 7 にある Reward 値（0 ～ +8）で与える eMCTS-Reward である。

比較対象に使った手法は表 1~3 を利用するベーシックストラテジー（Basic Strategy: BS）に基づく BS 法とカードカウンティング (Card Counting: CC) を併用した BS+CC である。カードカウンティングの手法は拡張モンテカルロ木探索とも併用できる。これを eMCTS-Reward + CC と表記する。

一方、プレイヤー数やデッキ数も様々に設定可能である。組み合わせが多数あるので 4 つに集約した。1 プレイヤー 1 デッキ、1 プレイヤー 6 デッキ、4 プレイヤー 1 デッキ、4 プレイヤー 6 デッキの 4 パターンに絞り込んだ。

カードカウンティングを使う場合、賭け金を変動させる必要がある。これには表 5 の「ランクに基づくカウント値 ±3」と表 6「ランニングカウント RC による賭け金の変動」を使用した。カードカウンティングを併用しない BS 法や eMCTS-WinLose 法、eMCTS-Reward 法の場合には、賭け金を 250 に固定にしてプレイさせた。

ディーラーがデッキセットを交換するタイミング、つまり、1 セットの終了タイミングは残りデッキに収納されているカード枚数が事前に設定した既定枚数を下回ったときとした。既定枚数は {(ディーラーを含むプレイヤー数 + 1) × 5} で与えた。“+1” の部分は余裕係数である。

拡張モンテカルロ木探索で使う UCB 値の計算式 (4) にある定数パラメータ C を 1.0, 2.0, 3.0, 4.0 と変化させて実験した。拡張モンテカルロ木探索の実行では他にもいくつかのパラメータを事前に設定する必要がある。プレイアウト数の上限を 10000 回に設定し、ノード展開閾値を 5 に設定して実験を行った。

ペイアウト率を算出するタイミングは特定セット数の終了時ではなく事前に設定したラウンド数の終了時とした。使用デッキ数が 1 デッキと 6 デッキの 2 つがあり、デッキ数への依存を極力排除するためである。1 プレイヤー 1 デッキ/1 プレイヤー 6 デッキ/4 プレイヤー 1 デッキ/4 プレイヤー 6 デッキの各パターンで最大ラウンド数（max rounds）をそれぞれ 5000/50000/8000/80000 として実験を行った。最大ラウンド数はすべての実験で同一に設定した。実験の公平性を担保するためである。少プレイヤーや小デッキ数のときにはラウンド数を少な目に設定し、多プレイヤーや大デッキ数のときにはラウンド数をその 10 倍に設定した。

プレイヤーが 1 人の場合のペイアウト率は式 (1) の通りである。一方、プレイヤーが 4 人の場合、<4 人の賭け金総額の合計 > に対する <4 人の払戻金総額の合計 > の比でペイアウト率を計算した。ブラックジャックは本来、プレイヤー 1 人が楽しむゲームである。しかし、提案法を特定のプレイヤー 1 人に適用する場合を想定すると、そのプレイヤーがテーブルに着座する場所まで厳密に考慮する必要がある。本研究ではそこまで考慮していない。個々のプレイヤーが個別に提案法で行動を選択する、と仮定している。これらの事情を勘案して数値実験では 4 プレイヤー 1 デッキと 4 プレイヤー 6 デッキの場合には、4 人の合計金額からペイアウト率を算出した。

6・2 実験結果と考察

まず、ベーシックストラテジー（BS）のみの実験結果を表 8 に示す。どのケースでもペイアウト率が 100%を超えている。

提案法であるプレイアウト報酬を勝敗で与える eMCTS-WinLose を UCB 計算式 (4) の定数パラメータ C を 1.0 にして、実験した。結果を表 9 に示す。4 ケースのすべてでペイアウト率が 95%にさえ届かなかった。囲碁と同じ勝敗 {+1, 0} による報酬では不完全情報ゲームであるブラックジャックには全く対応できないのかもしれない。

そこで、プレイアウト報酬を表 7 にある Reward 値（0 ～ +8）に切り替えて eMCTS-Reward の実験をした。プレイアウトの戻り値が勝敗 {+1, 0} ではなくより大きな整数になる。そのため、誤差項の役目を果たす第 2 項の係数パラメータを $C = 3.0$ と設定し直した。提案法 eMCTS-Reward（$C = 3.0$）の結果（表 10）は、4 ケースのすべてで明らかに eMCTS-WinLose（$C = 1.0$）よりも良くなっている。少なくとも 8%以上の改善に成功している。

続く実験はベーシックストラテジーとカードカウンティングの併用（BS+CC）である。ベーシックストラテジー（BS）を概ね上回る良好な結果が得られた。1 player, 6 decks のケースのみ劣っている。一覧を表 11 にまとめる。

最後に、提案法である拡張モンテカルロ木探索とカードカウンティングの併用（eMCTS-Reward+CC）（$C = 3.0$）である。拡張モンテカルロ木探索 eMCTS-Reward（$C = 3.0$）を単独で使用するよりも概ね良好な結果が得られた。表 12 にその一覧を示す。BS+CC のときと同様に 1 player, 6 decks のケースで勝てていない。

(BS), (BS+CC), (eMCTS-Reward+CC) の 3 つの手法についてペイアウト率だけを表 8, 表 11, 表 12 から抜き出してグラフ化してみる。比較した結果が図 4 である。群を抜いて高ペイアウト率を達成しているのが (BS+CC) である。図 4 から提案法の (eMCTS-Reward + CC) が明白に勝てているのは 1 プレイヤー 1 デッキの場合の BS 法に対してだけである。4 つのケースにおいてペイアウト率が最大となる手法を一覧表にまとめたのが表 13 である。残念ながら、この中に提案法はひとつも入っていない。グラフでも示した通り、BS+CC の圧勝である。

表 8 ベーシックストラテジー（BS）のみの結果

BS only	max rounds	wins	losses	draws	total bet	total profit	total refund	payout ratio
1 player, 1 deck	5000	2391	2412	287	1,385,500	49,750	1,435,250	103.59%
1 player, 6 decks	50000	23426	24304	3304	13,860,000	271,125	14,131,125	101.96%
4 players, 1 deck	8000	15018	15556	1978	8,855,750	204,875	9,060,625	102.31%
4 players, 6 decks	80000	148536	157022	20790	88,577,500	973,500	89,551,000	101.10%

表 9 拡張モンテカルロ木探索（eMCTS-WinLose）（$C = 1.0$）の結果

eMCTS-WinLose ($C = 1.0$)	max rounds	wins	losses	draws	total bet	total profit	total refund	payout ratio
1 player, 1 deck	5000	2317	2566	160	1,603,500	$-122,125$	1,481,375	92.38%
1 player, 6 decks	50000	22947	25820	1778	16,020,500	$-1,420,750$	14,599,750	91.13%
4 players, 1 deck	8000	14712	16525	966	10,253,750	$-864,125$	9,389,625	91.57%
4 players, 6 decks	80000	146208	165306	11108	102,279,750	$-9,178,375$	93,101,375	91.03%

表 10 拡張モンテカルロ木探索（eMCTS-Reward）（$C = 3.0$）の結果

eMCTS-Reward ($C = 3.0$)	max rounds	wins	losses	draws	total bet	total profit	total refund	payout ratio
1 player, 1 deck	5000	2175	2736	164	1,386,750	13,750	1,400,500	100.99%
1 player, 6 decks	50000	21700	27360	1656	13,768,000	86,750	13,854,750	100.63%
4 players, 1 deck	8000	13975	17363	1009	8,875,000	126,375	9,001,375	101.42%
4 players, 6 decks	80000	136750	176508	11011	88,001,750	$-457,000$	87,544,750	99.48%

表 11 ベーシックストラテジーとカードカウンティングの併用（BS+CC）の結果

BS+CC	max rounds	wins	losses	draws	total bet	total profit	total refund	payout ratio
1 player, 1 deck	5000	2295	2372	333	234,500	28,200	262,700	112.03%
1 player, 6 decks	50000	22801	23837	3362	3,578,200	28,300	3,606,500	100.79%
4 players, 1 deck	8000	14679	15346	1975	912,500	135,750	1,048,250	114.88%
4 players, 6 decks	80000	144745	153989	21266	5,998,700	322,100	6,320,800	105.37%

表 12 拡張モンテカルロ木探索とカードカウンティングの併用（eMCTS-Reward+CC）（$C = 3.0$）の結果

eMCTS-Reward + CC ($C = 3.0$)	max rounds	wins	losses	draws	total bet	total profit	total refund	payout ratio
1 player, 1 deck	5000	2128	2791	136	246,825	14,709	261,534	105.96%
1 player, 6 decks	50000	21572	27430	1692	455,865	$-8,396$	447,469	98.16%
4 players, 1 deck	8000	14029	17354	959	1,308,580	30,409	1,338,989	102.32%
4 players, 6 decks	80000	137154	176091	10953	19,113,980	196,109	19,310,089	101.03%

6・3 提案法の実行時間

数値実験の PC 環境は主として、MacBook Air, OS: macOS Big Sur Ver.11.6.8, CPU: 2.2GHz Dual Core Intel Core i7, RAM: DDR3 1600MHz 8GB, C/C++ Compiler: gcc Ver. 12.0.0 である。

拡張モンテカルロ木探索とカードカウンティングの併用（eMCTS-Reward + CC）によるペイアウト率の比較一覧を表 14 に示す。UCB 値の式 (4) の定数パラメータ C を $1.0 \to 2.0 \to 3.0 \to 4.0$ と徐々に大きくした場合の 4 パターンでのペイアウト率の比較である。プレイヤー数とデッキ数の組み合わせに対し、最も良好だったパラメータ C のペイアウト率を下線で強調している。例えば、1 player, 1 deck の数値実験では、定数パラメータを $C = 4.0$ に設定して、5000 ラウンド対戦するペイアウト率が 107.92%となり、最も良好であった。それに指標 (A) を与えている。この (A) 107.92%のペイアウト率を達成するための実行時間（execution time）が 39 分 2 秒であった

表 13 4 つのケースでペイアウト率が最大となる手法

	max round	payout ratio	method
1 player，1 deck	5000	112.03%	BS + CC
1 player，6 decks	50000	101.96%	BS only
4 players, 1 deck	8000	114.88%	BS + CC
4 players, 6 decks	80000	105.37%	BS + CC

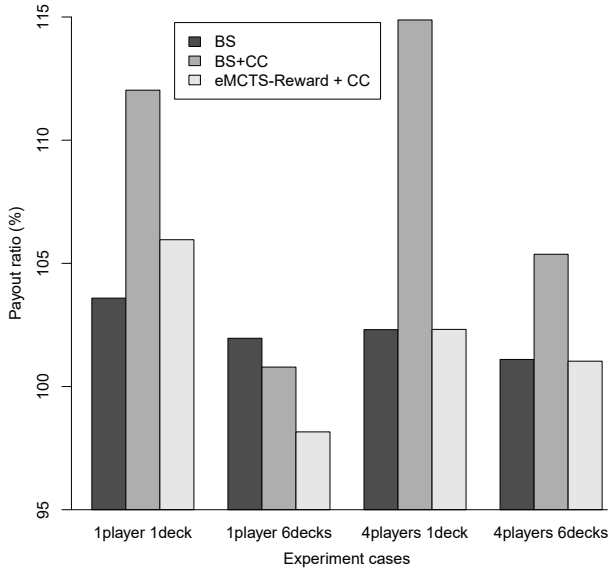

図 4 BS, BS+CC, eMCTS-Reward+CC のペイアウト率

ことを表 14 の最右欄で示している。4 players, 6 decks の数値実験では、80000 ラウンドを $C = 3.0$ にして対戦すると、ペイアウト率が 101.03%で最も良く、実行時間は 2 日 15 時間 52 分 6 秒であったことが分かる。秒に換算すると 229926 sec である。これを 1 プレイの時間に換算すると (80000 round × 4 play/round) で除して、約 0.71 sec/play の計算時間を要している。提案法の eMCTS-Reward + CC が 1 プレイに 0.71 秒を要す一方で、BS+CC はベーシックストラテジーの表を 3 つ使って表引きをするだけなので計算時間ほぼゼロに近い。

7. おわりに

不完全情報ゲームを攻略できるようにブラックジャックのゲーム木を構築し、内部ノードでもプレイアウトを可能にした拡張モンテカルロ木探索を提案した。プレイアウトの戻り値を使って UCB 値を更新する際は 2 つの方法を提案した。ひとつは囲碁の場合と同様、勝敗による {+1, 0} を利用する方法である。もう一つは勝敗に基づきブラックジャックに特化した 0 ～ +8 の報酬を与える方法である。数値実験から後者が優位であることを確認できた。

カードランクに基づくカウント値をランニングカウントに順次加算して賭け金を変動させるカードカウンティングを拡張モンテカルロ木探索と併用することも提案した。カードカウンティングはベーシックストラテジーとの相性が抜群であり、併用すれば強力な実践ツールとなる。それは数値実験からも確認することができた。

拡張モンテカルロ木探索とカードカウンティングを併用すれば、既存手法の BS+CC よりもペイアウト率の向上に寄与できると予想した。残念ながら、数値実験では期待したほどの結果は得られなかった。

今後の課題は、プレイアウトの戻り値をブラックジャック用に特化した報酬に改めるだけでなく、UCB 値の計算も不完全情報ゲームに対応した算出式に改める必要がある。さらに、ゲーム木や探索木の構成も一から見直す必要があるだろう。

謝　辞

本研究を遂行するにあたり、令和 3 年本学情報科学部情報科学科卒業の水田亮司氏（現在 航空自衛隊 勤務）と令和 4 年度同学部同学科在学の中山龍之介氏・櫻井一正氏（3 氏ともに当研究室所属）には多大な協力を得た。ここに謝意を表する。

参考文献

1) 大佐賀 猛, 中野秀俊, 吉川 毅, 杉本雅則: “不完全情報ゲームにおける適応的モンテカルロ木探索手法の提案”, 情報処理学会論文誌 数理モデル化と応用 Vol. 8, No. 1, pp. 38 – 44, 2015.
2) N. Brown and T. Sandholm: “Superhuman AI for heads-up no-limit poker: Libratus beats top professionals”, Science, Vol. 359, Issue 6374, pp. 418 – 424, 2018.
3) O. H. Woo and 金子知適: “GVG-AI のための Monte Carlo Tree Search の改善に関する研究”, The 22nd Game Programming Workshop 2017, pp. 56 – 63, 2017.
4) D. Silver, T. Hubert, et al.: “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”, *Science*, Vol. 362, pp. 1140 – 1144, 2018.
5) R. Coulom: “Computing Elo Ratings of Move Patterns in the Game of Go”, Computer Games Workshop 2007 (CGW 2007), Amsterdam, The Netherlands, 2007.
6) D. Silver, A. Huang, et al.: “Mastering the game of Go with deep neural networks and tree search”, *Nature*, Vol. 529, pp. 484 – 489, 2016.
7) D. Silver, J. Schrittwieser, et al.: “Mastering the

表 14 拡張モンテカルロ木探索とカードカウンティングの併用（eMCTS-Reward + CC）によるペイアウト率の比較と実行時間

eMCTS-Reward + CC	max round	C=1.0	C=2.0	C=3.0	C=4.0	execution time
1 player, 1 deck	5000	104.80%	104.26%	105.96%	(A) 107.92%	39m 2s (A)
1 player, 6 decks	50000	98.06%	99.43%	98.16%	(B) 99.86%	7h50m20s (B)
4 players, 1 deck	8000	101.80%	(C) 104.47%	102.32%	101.06%	6h38m22s (C)
4 players, 6 decks	80000	100.42%	98.56%	(D) 101.03%	99.13%	2day 15h52m 6s (D)

game of Go without human knowledge", *Nature*, Vol. 550, pp. 354 – 359, 2017.

8) 青野雅樹: Keras によるディープラーニング 実践テクニック & チューニング技法, 森北出版, 2017.
9) R. Munos: "From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning", *Foundations and Trends in Machine Learning*, Vol. 7, No. 1, pp. 1 – 129, 2014.
10) P. Auer, N. Cesa-Bianchi, and P. Fischer: "Finite-time Analysis of the Multiarmed Bandit Problem", *Machine Learning*, Vol. 47, Issues 2–3, pp. 235–256, 2002.
11) 齋藤隆浩: 新訂 ブラックジャック必勝法, データハウス, 1999.
12) R. Morris, 田崎涼子 訳: カードカウンティング入門, パンローリング, 2012.
13) E. O. Thorpe: Beat the Dealer: A Winning Strategy for the Game of Twenty-One, Vintage Books, NY, 1963.
14) L. Kocsis and C. Szepesvári: "Bandit based Monte-Carlo Planning", Proceedings of the 17th European Conference on Machine Learning (ECML 2006), pp. 282 – 293, 2006.

（受理　令和 5 年 3 月 20 日）