卒業研究報告書

題目

ジャンケンを題材にしたゲームの洗練法

指導教員

石水　隆　講師

報告者

12-1-037-0101

加藤　聡士

近畿大学理工学部情報学科

平成28年1月31日提出

概要

じゃんけんは偶然性に多くを支配されるゲームであるという特性から、しばしば確率の問題で使われることがある。一方、勝敗について戦略を練る余地が無いため、ゲームとして見た場合の面白みは少ないと言える。ゲームでの面白さは自由度 B(ボードゲームでは平均合法手数)と時間 D (開始から終了までの平均手数)を用いて、B^D および$\sqrt{B}/D$、で評価することが提案されている 1)。B^Dはゲーム木の大きさを表し、ゲームの難しさを表す要素である4)。$\sqrt{B}/D$は洗練度と呼ばれる値であり、洗練されたゲームとされるチェス0.074、将棋では0.078となる 4)。 本研究で、確率依存型の単純なゲームのじゃんけんに、いくつかの特殊ルールを追加し、ゲームとしてより面白くなることを追求する。また、そのルールを B^D および$\sqrt{B}/D$で評価し、この2つの評価値がゲームとしての面白さを正しく評価しているか検証する。

目次

1. 序論

1.1. じゃんけんの概要

じゃんけんのルールを知らない人がいないと思うが説明をする。じゃんけんは一般的には２人で行うゲームであるが、ときには２人以上で行うこともある。本研究では 2 人でのじゃんけんに限定して考える。じゃんけんは３通りの手が存在する。３通りの手の名称は、グー・チョキ・パーである。 それぞれのプレイヤは３通りの手の中から1つを選択し、掛け声と同時に両者自分の選択した手を同時に見せ勝敗を決める。図１に勝敗の有向グラフを示す。

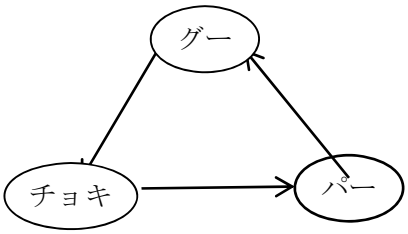

図1 じゃんけんの有向グラフ

- グーはチョキに勝ち、パーに負ける。
- チョキはパーに勝ち、グーに負ける。
- パーはグーに勝ち、チョキに負ける。
- お互い同じ手の場合、あいことなる。

じゃんけんの3つの手は、それぞれ一つの手には勝ち、もう一方の手には負ける。同じ手とはあいことなる。このように、じゃんけんゲームには絶対的に優位な手というものが存在しておらず、それぞれの手は3すくみの関係になっている。

1.2. じゃんけんに関する既知の結果

10ラウンドのラウンドマッチじゃんけんを基本ゲームとして、特別ルールを追加することでより洗練されたゲームに調整できるかどうかを試みている。そして、2ラウンド連続勝利を3回で勝利 になるという特別ルールが面白いゲームと述べている。[1]本研究では、別のラウンドマッチじゃんけんや特別ルールで洗練度を求め、ほかにも複雑度に注目しより面白いゲームにならないかを求めている。

1.3. ゲームの洗練度

ゲームの面白さを表す指標として、洗練度と複雑度がある。ゲームの自由度 B (ボードゲームでは平均合法手数)と時間 D(開始から終了までの平均手数)を用いて、洗練度は$\sqrt{B}/D$、複雑度 B^D で表される[4][9]。洗練度はゲームの勝敗を決める「スキル」と「チャンス」のバランスを表すパラメタであり、複雑度はゲーム中に起こりえる局面数を表すパラメタである。一般に面白いゲームとされているチェス、将棋の場合、自由度 Bはそれぞれ 35, 80 程度であり、時間 Dはそれぞれ 80, 115 程度とされている[9]。よって、チェス・将棋の洗練度はそれぞれ 0.074, 0.078 であり、複雑度はそれぞれ $35^{80} = 3.35*10^{125}$, $80^{115} = 7.15*10^{218}$ となる。

1.4. 本研究の目的

本研究では、簡単なゲームのじゃんけんにいくつかのルールを追加し、面白いと言われているチェスや将棋のゲームの洗練度に近いルールを見つけ出す。また、評価値複雑度および洗練度がゲームとしての面白さを正しく評価しているか検証する。

1.5. 本報告書の構成

2節では、本研究でラウンドマッチ、特別ルールについて記述する。第3節では、本研究の結果および考察を記述する。第４節結論および今後の課題を記述する。

2. 研究内容

通常の 2 人で行うじゃんけんの場合、勝ち、負け、引き分けの発生確率は統計的に言えば1/3ずつである。この発生確率は偶然のみによって決まり、プレイヤの戦略の入る余地が無い。そこで本研究ではじゃんけんに戦略性を加えるため、ラウンドマッチじゃんけんを提案する。これは、じゃんけんに得点に関する特別ルールを加えた上でラウンドマッチにすることでより戦略性を持たせたゲームである。

2.1. ラウンドマッチじゃんけん

通常の2人で行うじゃんけんの場合、1回限りの対戦では、勝ち、負け、引き分けの発生確率は統計的に言えば1/3ずつとなる。１回限り場合、勝敗は運に作用される。そこで、本研究では、じゃんけんに戦略を取り入れようとするために、じゃんけんを複数回行うラウンドマッチじゃんけんを提案する。じゃんけんを何回も繰り返して行っていくなかで、 1回ごとの勝負の結果や、相手の手の履歴などを用いる事で、勝負における戦略というものを考える事ができる。以下にラウンドマッチじゃんけんのルールを記載する。

- 勝利するごとにポイントが加算され、決められたポイントに達すると勝利とする。
- 出した手によって獲得ポイントが変わる。
- 連続で勝利するとボーナスポイントが得られる。
- 連続勝利中引き分けになると連続勝利は途切れる。

このような特別ルールによって単なる確率型ゲームでは起こり難い逆転の発生を期待出来る。これによって,シーソーゲームになる確率が増え、ゲームとしてより魅力的になると考えられる。

2.2. 計算機実験

本研究では、ラウンドマッチじゃんけんのゲーム性を検証するために、計算機実験を行った。計算機実験で用いたプログラムのソースを付録に示す。

計算機実験は以下の条件で行った。

1.勝利ポイント20点先取とする。

2.両プレイヤとも各手に付加された得点に比例した確率で出す。

また、連続勝利によるボーナスは表1に示す7通り、出した手による獲得ポイントは表2に示す6通りの計42通りで実験を行った。これの目的として、何回もラウンドを行わないと終了しないものや、1回で決着がつくもので選んだ。

表1　連勝数によるボーナス

a	連勝ボーナス無し
b	2連勝以上で +1
c	2連勝以上で +10
d	3連勝以上で +3
e	3連勝以上で +10
f	2連勝で+1, 3連勝で+2, 4連勝で+3, 5連勝で+4, ...
g	2連勝で*2, 3 連勝で*3, 4 連勝で*4, 5 連勝で*5, ...

表2　出した手によるボーナス

1	グー1 チョキ1 パー1
2	グー2 チョキ1 パー1
3	グー3 チョキ2 パー1
4	グー5 チョキ3 パー1
5	グー10 チョキ1 パー1
6	グー20 チョキ10 パー1

3. 結果および考察

本研究では、2.2節で述べた条件で各特別ルールの組み合わせに対して各10000回の計算機実験を実施した。

表3. 5に各条件での平均ラウンド数 D および洗練度$\sqrt{B}/D$、を示す。表5より、ルール(3:グー3 チョキ2 パー1,b:2連勝以上で +1) および (グー3 チョキ2 パー1,d:3連勝以上で +3) の 組み合わせが将棋やチェスの洗練度 0.074と0.078 に近いことが示される。出した手によるボーナスや連勝数によるボーナスの点数が高いものは洗練度0.07より低く、逆に出した手によるボーナスや連勝数によるボーナスの点数が低いものは洗練度0.07より高くなることが表5からわかる。この結果より、ただ何回もじゃんけんをしたり、1回で決着がついてしまったりするルールでは面白いとされる洗練度の値にならないことがわかる。

B=3で固定されているので、洗練度の値は Dにのみ依存する。$\sqrt{B}/D$=0.07となるのは D=24のときである。つまり、戦略性があうが無かろうが、じゃんけんを延々24回するというルールにしたときが最も洗練度が0.07に近くなる。しかし、じゃんけんを繰り返しするだけで面白いとはいえない。すなわち、洗練度の値が必ずしもゲームとしての面白さを表しているとは言えない。面白いゲームとするためには、ある程度の特別ルールを足し、単なる確率型ゲームでは起こり難い逆転の出来るゲールとする必要があるだろう。

一方、複雑度 B^D については、チェスと将棋の複雑度はそれぞれ $35^{80} = 3.35*10^{125}$, $80^{115} = 7.15*10^{218}$である。表4より、2つの複雑度の値に近いものは1つも無い事がわかる。この結果より、本研究での特別ルールはチェスや将棋に比べゲームの複雑さがないことがわかった。

表3　各ルールでの平均終了ラウンド数 D

	1	2	3	4	5	6
a	55.24	40.80	26.67	17.29	11.62	5.06
b	40.13	31.74	22.35	15.44	11.25	4.88
c	12.07	11.79	10.47	8.59	6.64	4.46
d	39.06	31.47	22.46	15.56	11.39	5.04
e	26.23	18.06	17.91	13.55	10.06	4.94
f	35.45	28.99	21.06	14.90	10.96	4.89
g	35.21	26.58	17.95	12.26	9.41	4.,67

表4　各ルールでのゲームの複雑度 B^D

	1	2	3	4	5	6
a	$2.27*10^{26}$	$2.90*10^{19}$	$5.30*10^{12}$	$1.77*10^{8}$	$3.50*10^{5}$	$2.59*10^{2}$
b	$1.40*10^{19}$	$1.39*10^{15}$	$4.60*10^{10}$	$2.32*10^{7}$	$2.33*10^{5}$	$2.12*10^{2}$
c	$5.73*10^{5}$	$4.21*10^{5}$	$9.89*10^{4}$	$1.25*10^{4}$	$1.47*10^{3}$	$1.34*10^{2}$
d	$4.32*10^{18}$	$1.03*10^{15}$	$5.20*10^{10}$	$2.65*10^{7}$	$2.71*10^{5}$	$2.53*10^{2}$
e	$3.27*10^{12}$	$4.13*10^{8}$	$3.50*10^{8}$	$2.91*10^{6}$	$6.30*10^{4}$	$2.27*10^{2}$
f	$8.20*10^{16}$	$6.78*10^{13}$	$1.11*10^{10}$	$1.28*10^{7}$	$1.69*10^{5}$	$2.15*10^{2}$
g	$6.30*10^{16}$	$4.80*10^{12}$	$3.66*10^{8}$	$7.07*10^{5}$	$3.08*10^{4}$	1.69*10

表5　各ルールでのゲームの洗練度$\sqrt{B}/D$、

	1	2	3	4	5	6
a	0.031	0.038	0.064	0.100	0.149	0.342
b	0.043	0.054	0.077	0.112	0.153	0.354
c	0.143	0.153	0.165	0.201	0.260	0.388
d	0.043	0.055	0.077	0.111	0.152	0.343
e	0.066	0.095	0.096	0.127	0.172	0.350
f	0.048	0.059	0.082	0.116	0.158	0.354
g	0.049	0.065	0.096	0.141	0.184	0.370

4. 結論・今後の課題

本研究では、じゃんけんをより面白くするためにラウンドじゃんけんを提案した。本研究では、じゃんけんを面白くするために洗練度に着目し、それが面白い値となるルールの追加を行った。

本研究では、ラウンドマッチじゃんけんに出した手によるボーナスや連勝数によるボーナスの特別ルールを加え、洗練度を求めた。結果、24ラウンドで終了する特別ルールを追加するのが面白いとされる洗練度の値になるとわかった。だが、第3章で考察したように、洗練度の値が0.07だからといって面白いとは限らない。したがって新たな特別ルールを追加するにしても、その洗練度が0.07することを目指すのではなく、面白いゲームとなることを目指す必要がある。また、その場合の洗練度どのような値になるか解析し、面白さと洗練度の関係を調べる必要がある。

本研究のじゃんけんでは、複雑度の値が低いという結果から、面白く無いゲームと判断してはならない。だが、複雑度の値が高いから面白いゲームと判断するのもよくない。複雑度の値が高くするために新たなルールを足しても、ただプレイがしにくくなるだけである。この結果より洗練度と複雑度の関係性を調べることが必要である。また、数値で面白いかを判断するのではなく、実際に人にじゃんけんをしてもらい面白いかをアンケートを取り、人が思っている面白いと洗練度が近いものか検証すべきである。

謝辞

本研究を行うにあたって、近畿大学理工学部情報学科情報論理工学研究室の石水 隆講師には大変お世話になりました。研究に関する指摘やサポート、アドバイスなど適切な指導をしていただいたことを、ここに感謝をこめて記します。

参考文献

[1] 柏木理志, 飯田弘之: ゲームの洗練法 -じゃんけんを題 材として- , 情報処理学会研究報告 2003-GI-010, pp.9- 13(2003), http://id.nii.ac.jp/1001/00058569/

[2] D.Billing. Thoughts on RoShamBo, ICGA Journal, Vol23,NO.1,pp.3-8(2000)

[3] 漆間幸雄,飯田弘之：ゲーム洗練度の理論とポーカー, ゲームプログラミングワークショップ2005論文集 Vol.2005, No.15,pp.138-141 (2005), http://id.nii.ac.jp/1001/00097591/

[4] 佐々木　宣介,橋本　剛,梶原　羊一郎,飯田弘之：チェスライクゲームにおける普遍的指標, 情報処理学会　研究報告ゲーム情報学, Vol.1999-GI-001, No.53, pp.91-98 (1999), http://id.nii.ac.jp/1001/00058670/

[5] 伊藤大雄, 一般化ジャンケン,オペレーションズ・リサーチ, Vol.18, No.3, pp.156-160, (2013), http://www.orsj.or.jp/archive2/or58-03/or58_3_156.pdf

[6] 山下将臣, じゃんけんゲームに対する遺伝的アプローチ, 高知工科大学 情報システム工学科 平成20年度 学士学位論文, (2009), http://www.kochi-tech.ac.jp/library/ron/2008/2008info/1090396.pdf

[7] 服部太輔, 細江政範,　石黒友一, ジャンケンの文化的側面と数理解析, 南山大学 数理情報学部 数理科学科 2006年度卒業研究, (2007), http://www.seto.nanzan-u.ac.jp/st/gr-thesis/ms/2006/osaki/03mm015.pdf

[8] 牧野泰裕, 西野順二, 小高知宏, 小倉久和, 繰り返し対戦型じゃんけんにおける戦略の特徴, 福井大学 工学部 研究報告 第45巻 第1号, pp.71-79, (1997), http://repo.flib.u-fukui.ac.jp/dspace/bitstream/10098/3425/1/AN00215401-045-01-007.pdf

[9] 佐々木 宣介,飯田 弘之：将棋種の歴史的変遷の解析, 情報処理学会論文誌 Vol.43, No.10, pp.2990-2997, (2002), http://id.nii.ac.jp/1001/00011455/

付録

本研究で作成したじゃんけんプログラムのソースを以下に示す。

```
iimport java.util.Scanner;
import java.util.Random;

public class Roshambo {
	static final int GU = 0; // グー
	static final int CH = 1; // チョキ
	static final int PA = 2; // パー
	static final int WIN = 1; // 勝ち
	static final int LOSE = -1; // 負け
	static final int DROW = 0; // 引き分け
	static final int[][] isWin = { { DROW, WIN, LOSE }, // 勝敗判定
				{ LOSE, DROW, WIN }, { WIN, LOSE, DROW } };
	static final String[] handToString = { "グー", "チョキ", "パー" }; // 手の文字列表現
	int round; // 対戦ラウンド数
	int[] score; // 得点
	int[] victries; // 勝利数
	int[] svictries; // 連勝数
	int maxRound; // 最大ラウンド数
	int boundScore; // 勝利となる規定得点
	int limitDiffScore; // 勝利となる得点差
	int[] handBonus; // 出した手によるボーナス
	int count = 0;
	int round_cnt = 0;
	static final int handBonusType = 1; // 出した手によるボーナスのタイプ
	/*
	 * 0:グー1 チョキ1 パー1
	 * 1:グー2 チョキ1 パー1
	 * 2:グー3 チョキ2 パー1
	 * 3:グー5 チョキ3 パー1
	 * 4:グー10 チョキ1 パー1
	 * 5:グー20 チョキ10 パー1
	 */
	static final int comboBonusType = 2; // 連勝数によるボーナスタイプ
	/*
	 * 0:連勝ボーナス無し
	 * 1:2連勝以上で+1
	 * 2:2連勝以上で+10
	 * 3:3連勝以上で+3
	 * 4:3連勝以上で+10
	 * 5:2連勝で+1,3連勝で+2, 4連勝で+3, 5連勝で+4, ...
	 * 6:2連勝で*2, 3連勝で*3, 4連勝で*4, 5連勝で*5, ...
	 */

	static final int gameSetType = 0; // 終了条件のタイプ
	/*
	 * 0:20点先取
	 */

	static final int comLevel = 0; // COM の戦略レベル
	/*
	 * 0:各手に付加された得点に比例して出す
	 */

	int[] weight; // 各手を出す確率に付加する重み

	Scanner keyBoardScanner; // キーボードスキャナ
	Random rnd; // 乱数発生用

	/**
	 * コンストラクタ 各初期値をセットする
	 */
	Roshambo() {
		round = 0; // ラウンド数
		score = new int[2]; // 得点
		score[0] = 0;
		score[1] = 0;
		victries = new int[2]; // 勝利数
		victries[0] = 0;

		victries[1] = 0;
		svictries = new int[2]; // 連勝数
		svictries[0] = 0;
		svictries[1] = 0;
		handBonus = new int[3];
```

```
		switch (handBonusType) { // 出した手によるボーナス
		case 0: // グー1 チョキ1 パー1
				handBonus[GU] = 1;
				handBonus[CH] = 1;
				handBonus[PA] = 1;
				break;
		case 1: // グー2 チョキ1 パー1
				handBonus[GU] = 2;
				handBonus[CH] = 1;
				handBonus[PA] = 1;
				break;
		case 2: // グー3 チョキ2 パー1
				handBonus[GU] = 3;
				handBonus[CH] = 2;
				handBonus[PA] = 1;
				break;
		case 3: // グー5 チョキ3 パー1
				handBonus[GU] = 5;
				handBonus[CH] = 3;
				handBonus[PA] = 1;
				break;
		case 4: // グー10 チョキ1 パー1
				handBonus[GU] = 10;
				handBonus[CH] = 1;
				handBonus[PA] = 1;
				break;
		case 5: // グー20 チョキ10 パー1
				handBonus[GU] = 20;
				handBonus[CH] = 10;
				handBonus[PA] = 1;
				break;

		}
		switch (gameSetType) { // ゲーム終了条件

		case 0: // 20点先取
				maxRound = Integer.MAX_VALUE;
				boundScore = 20;
				limitDiffScore = Integer.MAX_VALUE;
				break;


		}
		weight = new int[3];
		setWeight(comLevel); // 各手を出す重みを設定する

		keyBoardScanner = new Scanner(System.in); // 入力をキーボードに設定
		long seed = System.currentTimeMillis(); // 現在時刻から乱数の種を生成
		rnd = new Random(seed); // 乱数発生用
}

/**
 * メインメソッド
 */
public static void main(String[] args) {
		Roshambo roshambo = new Roshambo(); // インスタンス生成
		roshambo.play(); // ゲーム開始
}

/**
 * ゲームを開始する
 */
void play() {
		int[] hand = new int[2];
		for (int game = 1; game < 10001; game++) {
				while (round < maxRound // 最大ラウンド数以内
						&& score[0] <= boundScore
						&& score[1] <= boundScore // 規定得点以下
						&& (score[0] - score[1]) <= limitDiffScore
						&& (score[1] - score[0]) <= limitDiffScore) { // 規定得点差以下

						showResult(); // 現在の結果表示
						hand[0] = com(comLevel); // プレイヤーの手入力
						hand[1] = com(comLevel); // COM の手選択
						//System.out.println
("あなた：" + handToString[hand[0]] + "　私：" + handToString[hand[1]]);
						int result = isWin[hand[0]][hand[1]]; // 勝敗判定
						int winner = -1; // 勝者
						if (result == WIN) {
								winner = 0;
```

```
		} else if (result == LOSE) {
			winner = 1;
		} else {
		}
		if (result == WIN || result == LOSE) {
			++victries[winner]; // 勝利数増加
			++svictries[winner]; // 連勝数増加
			svictries[1 - winner] = 0; // 連勝数リセット
			int handScore = handBonus[hand[winner]];
			int comboScore = 0;
			switch (comboBonusType) { // 連勝ボーナスに応じた得点追加
			case 0: // 連勝ボーナス無し
				score[winner] += handScore;
				break;
			case 1: // 2連勝以上で+1
				if (svictries[winner] >= 2) {
					score[winner] += (handScore + 1);
				} else {
					score[winner] += handScore;
				}
				break;
			case 2: // 2連勝以上で+10
				if (svictries[winner] >= 2) {
					score[winner] += (handScore + 10);
				} else {
					score[winner] += handScore;
				}
				break;
			case 3: // 3連勝以上で+3
				if (svictries[winner] >= 3) {
					score[winner] += (handScore + 3);
				} else {
					score[winner] += handScore;
				}
				break;
			case 4: // 3連勝以上で+10
				if (svictries[winner] >= 3) {
					score[winner] += (handScore + 10);
				} else {
					score[winner] += handScore;
				}
				break;

			case 5: // 2連勝で+1, 3連勝で+2, 4連勝で+3, 5連勝で+4, ...
				if (svictries[winner] >= 2) {
					comboScore = svictries[winner] - 1;
					score[winner] += (handScore + comboScore);
				} else {
					score[winner] += handScore;
				}
				break;

			case 6: // 2連勝で*2, 3連勝で*3, 4連勝で*4, 5連勝で*5, ...
				if (svictries[winner] >= 2) {
					comboScore = svictries[winner];
					score[winner] += (handScore * comboScore);
				} else {
					score[winner] += handScore;
				}
				break;

			default: // 連勝ボーナス無し
				score[winner] += handScore;
				break;
			}
		} else { // 引き分けの場合
			svictries[0] = 0; // 連勝数リセット
			svictries[1] = 0;
		}
		++round;
}

showResult(); // 結果表示
round_cnt += round;
count = round_cnt / 10000;
System.out.println(game + "ゲーム目終了.　現在のラウンド数:" + round_cnt);
System.out.println("平均" + count + "ラウンド");
round = 0;
victries[0] = 0;
victries[1] = 0;
score[0] = 0;
```

```
                        score[1] = 0;
                        svictries[0] = 0;
                        svictries[1] = 0;

                }
        }

        /**
         * 結果表示
         */
        void showResult() {
        }

        int com(int level) {
                int hand = -1;
                switch (level) { // COM のレベルに応じた手を選択
                case 0: // ランダムに選択
                        hand = rnd.nextInt(3);
                        break;
                }
                return hand;
        }

        void setWeight(int level) {
                switch (level) {
                case 0: // 各手に付加された得点に比例して出す
                        weight[GU] = handBonus[GU];
                        weight[CH] = weight[GU] + handBonus[CH];
                        weight[PA] = weight[CH] + handBonus[PA];
                        break;

                }
        }
}
```