注意
プレゼン用のスライドではないため
過剰に文章が多くなっている点と
フォントや文字の大きさが統一されていない点には
目を瞑って下さい
1種の参考程度に見て下さい
また名前は偽名を使用しています

# 対戦型ゲームの思考ルーチン
# ～3つの戦術を駆使する～

黒田にゃんた　白雪えなぴ　氷川ぺんた　犬塚わん

# 目次

# 実験プロジェクトの計画概要

- 実験遂行スケジュール

  6月09日:コードの改良

  6月16日:第1回コンテスト、中間レポート作成、コンテスト結果をふまえた改良

  6月23日:コンテスト結果をふまえたコードの改良、最終レポートの作成

  6月30日:第2回コンテスト、最終レポートの作成

  （すべての日程とも全員参加した）

- 遂行時の問題点

  javaの理解に時間がかかる ⇒ Copilotを活用しながら理解

⇒最初はそれぞれでプログラムを作り、案を持ち寄り統合。

途中からは統合した1つのプログラムを改善していく。

# 実験プロジェクトの計画概要

- 役割分担

  黒川にゃんた：パワーポイント責任者・修正点アルゴリズム思案

  白雪えなぴ：プログラミング・新分析アルゴリズム思案

  氷川ぺんた：プログラミング長・ログ解析と評価・新分析アルゴリズム思案

  犬塚わん：ログ解析と評価・問題点分析及び改善点アルゴリズム思案

（もちろん全員がお互いの役割を助け合いながらコーディングも発表も行った）

# アルゴリズムの概要(1)

## 第一回コンテストのログ解析に基づいた仮説

ほぼすべての班が3パターンの賭け方に分類される

①パワー型（1or5ドルのみを賭ける）
②ディフェンス型（1or2ドルのみを賭ける）
③バランス型（1～5ドルを賭ける）

# アルゴリズムの概要(2)

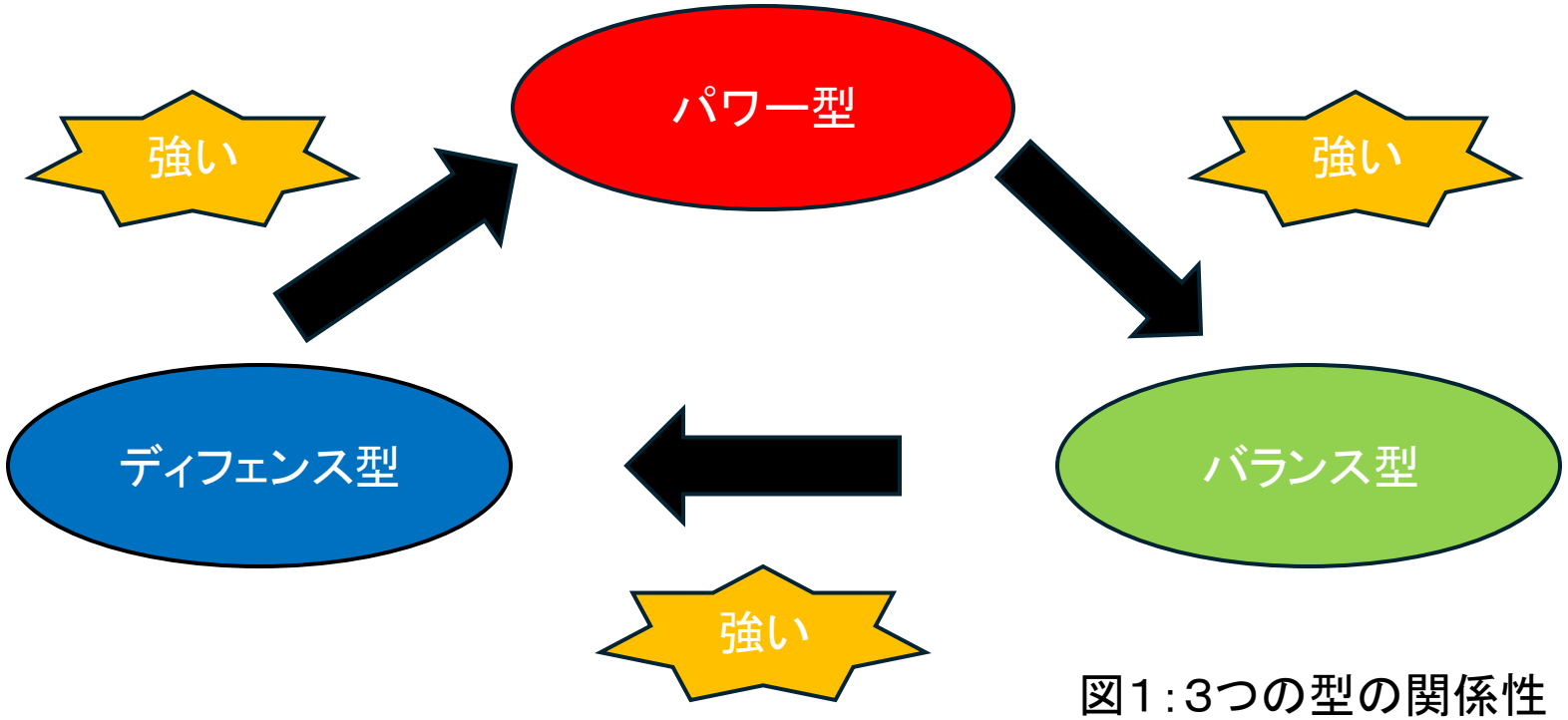


図1:3つの型の関係性

パワー型はバランス型に強く、バランス型はディフェンス型に強く、ディフェンス型はパワー型に強い
→3すくみの関係(ログ解析によって得られた仮説)
・相手がバランス型だとカードの予測がしやすいため低リスクで5ドルをかけられるパワー型が有利
・相手がディフェンス型だと相手が小額しか賭けないので高い金額を安全に賭けやすいバランス型が有利
・相手がパワー型だと相手の大きな賭けに乗らず損失を最小化するディフェンス型が有利

# アルゴリズムの概要
# (3すくみであると判断した理由)

第1回コンテストは以下の表1のように私たち2班が最も勝率が高かった
私たちは期待値計算に基づいたパワー型で戦っており
1.3.5班はそれぞれの賭けるドル数に差はあれどバランス型であった
唯一負けた4班が1.2ドルを賭けてきておりこれを私たちはディフェンス型と定義し
この4班が私たち2班以外には負けていたことから3すくみであると判断した

| チーム1~5 | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| グループ | 1 | | | | 2 | | | | 3 | | | | 4 | | | | 5 | | | | -- | | | | ポイント | 勝利数 |
| 1 | | | | | × | 0 | - | 3 | ○ | 2 | - | 1 | × | 1 | - | 2 | ○ | 2 | - | 1 | | | - | | 6 | 2 |
| 2 | ○ | 3 | - | 0 | | | | | ○ | 3 | - | 0 | × | 0 | - | 3 | ○ | 2 | - | 1 | | | - | | 9 | 3 |
| 3 | × | 1 | - | 2 | × | 0 | - | 3 | | | | | ○ | 2 | - | 1 | × | 0 | - | 3 | | | - | | 3 | 1 |
| 4 | ○ | 2 | - | 1 | ○ | 3 | - | 0 | × | 1 | - | 2 | | | | | × | 1 | - | 2 | | | - | | 6 | 2 |
| 5 | × | 1 | - | 2 | × | 1 | - | 2 | ○ | 3 | - | 0 | ○ | 2 | - | 1 | | | | | | | - | | 6 | 2 |
| -- | | | - | | | | - | | | | - | | | | - | | | | - | | | | | | 0 | 0 |

表1:第1回コンテスト結果

# アルゴリズムの特徴-1

コールorドロップについて

相手の掛け金ごとに、自分のカードの数値を配列で保存

→（自分のカードの予測値）= 0.7×（中央値）+ 0.3 ×（平均値）

```
        // 予測値: 安全重視で中央値*0.7 + 平均値*0.3
        double pred_score = pred_med * 0.7 + pred_avg * 0.3;
        int pred_card = (int)Math.round(pred_score);

        // 相手のカードが予測値より強い場合はドロップ
        if (compareCards(current.opponent_card, pred_card) > 0) {
            decision = "d";
        } else {
            decision = "c";
        }

        return decision;
    }
```

コード1

# アルゴリズムの特徴-2

200回まではディフェンス型（1，2ドルを賭ける）

⇒自分がローリスクで、相手の賭け方を学ぶため。

・相手のカードが2以上4以下のとき、2ドルを賭ける。

・相手のカードが5以上のとき、1ドルを賭ける。

```
//  2以上4以下には2をかけて5以上には1
if (current.opponent_card >= 2 && current.opponent_card <= 4) {
    b = 2;
} else {
    b = 1;
}
break;
```

コード2

# アルゴリズムの特徴-2

## ～ディフェンス型で相手のカードが2,3,4のとき5ドルをかける理由～

【相手が2,3,4・・・kの時だけ5ドルベット】

自分が勝つ確率 p＝（自分の手札が相手より強い枚数）/ 12であり、

得られる金額の期待値Eはp×2+(1-p)×(-2)である

実際に計算すると、期待値はk = 2, 3, 4, 5, 6, で正、それ以外は0以下となる

しかし、実際に多くの対戦を通して勝率が良かったのはk = 2, 3, 4, 5 の時であったため、今回はこの賭け方を採用した

# アルゴリズム特徴-3

200回を超えたら途中で賭け方を変える

1.ディフェンス型（元の1，2ドルを賭ける方法）に変更

⇒相手の1ドルベットが40%を超えているかつ2ドルより5ドルベットのほうが多いとき(相手はパワー型)

2.パワー型（1または5ドルを賭ける方法）に変更

⇒相手の1ドルベットが40%に満たないとき（相手はバランス型）

3.バランス型（1から5ドルを賭ける方法）に変更

⇒相手の1ドルベットが40%を超えているかつ5ドルより2ドルベットのほうが多いとき（相手はディフェンス型）

# アルゴリズム特徴-3

```
        if (rate1 < 0.4) {
            strategyType = STRATEGY_POWER;
        } else {
            if (count5 > count2) {
                strategyType = STRATEGY_DEFENCE;
            } else {
                strategyType = STRATEGY_BALANCE;
            }
        }
    }
```

コード3

# アルゴリズムの特徴-4

## ～パワー型～

相手のカードが2,3,4,5の時、相手の賭け金に関係なく5ドル賭け、それ以外では1ドル賭ける。

```
// 賭け金の決定（条件に応じて書き換え済み）
public String bid() {

    // 賭ける前にゲーム履歴情報を更新する
    HistoryUpdate();

    int b = 1; // デフォルトは1ドル

    // 相手のカードが2,3,4の時は5ドル、それ以外は1ドル
    if (current.opponent_card == 2 || current.opponent_card == 3 || current.opponent_card == 4 ||current.opponent_card == 5) {
        b = 5;
    } else {
        b = 1;
    }
```

コード4

# アルゴリズムの特徴-4

## ～パワー型で相手のカードが2,3,4,5のとき5ドルをかける理由～

【相手が2,3,4・・・kの時だけ5ドルベット】

自分が勝つ確率 p＝（自分の手札が相手より強い枚数）/ 12であり、

得られる金額の期待値Eはp×5+(1-p)×(-5)である

実際に計算すると、期待値はk=2,3,4,5で正、それ以外は0以下となる

この結果とリスク管理を考えると2, 3, 4, 5の時に5ドルかけるべきだとわかる

# アルゴリズム特徴-5

～バランス型～

相手のカードが2，3の時5ドル
相手のカードが4，5の時4ドル
相手のカードが6，7，8の時3ドル
相手のカードが9，10，11の時2ドル
それ以外の時は1ドル　賭ける

```
case STRATEGY_BALANCE:
    // バランス型:
    // 2,3のとき5ベット
    if (current.opponent_card == 2 || current.opponent_card == 3) {
        b = 5;
    }
    // 4,5のとき4ベット
    else if (current.opponent_card == 4 || current.opponent_card == 5) {
        b = 4;
    }
    // 6,7,8のとき3ベット
    else if (current.opponent_card == 6 || current.opponent_card == 7 || current.opponent_card == 8) {
        b = 3;
    }
    // 9,10,11のとき2ベット
    else if (current.opponent_card == 9 || current.opponent_card == 10 || current.opponent_card == 11) {
        b = 2;
    }
    // それ以外は1ベット
    else {
        b = 1;
    }
    break;
```

コード5

# アルゴリズムの特徴-5

## ～バランス型で相手のカードをこのような賭け方にした理由～

パワー型、ディフェンス型の期待値計算と同様にして、

計算を1ドルから5ドルそれぞれについても行う。すると以下の賭け方が最適だとわかる

相手のカードが2，3の時5ドル
相手のカードが4，5の時4ドル
相手のカードが6，7，8の時3ドル
相手のカードが9，10，11の時2ドル
それ以外の時は1ドル　賭ける

# 実装プログラムの特徴

・if文ではなくswitch文を使ったこと
(チームメンバーで共同開発する上での可読性の向上)

・対戦履歴を自分のカードの数値予測への利用だけではなく、
自分自身の賭け方の変更にも利用した

・対戦履歴を参照して自分のカードの数字を予測する際に、平均
値よりも中央値に重みを付けた
→履歴が少なかったり、外れ値があっても影響を受けにくい

# コンテスト結果の評価

- **アルゴリズムから予想される結果**
  相手の戦術に対して、それに合った戦術を利用して戦って勝つ

- **コンテスト結果の分析（2勝2敗）**
  相手の賭け金から、自分のカードの予測をうまくできなかった

# コンテスト結果の評価.1

**1班 (0-3で敗北)**

・Ourcard 2or14 TheyBet 5 : 自分のカードをうまく予測できない

・Ourcard not 2or14 TheyBet 1 : 幅が広く予測できない

相手はパワー型で自分はディフェンス型のため

仮説通りなら有利なはずなのに敗北

→ 自分のカードを予測できず運のコールorドロップになっていた

**3班 (3-0で勝利)**

・Ourcard from2to7 TheyBet 5

・Ourcard notfrom2to7 TheyBet 1

→ 3班はパワー型 and 3試合とも200回以内に勝利(こちらはディフェンス型)

仮説通りにパワー型に対してディフェンス型が完封したと言える

# コンテスト結果の評価.2

**４班(1−2で敗北)**

1戦目：ディフェンス型 →(200回目・相手24ドル／自分36ドル)
→バランス型 →232回目で敗北
2戦目：ディフェンス型のみ →105回目で敗北
3戦目：ディフェンス型 →(200回目：相手20ドル／自分40ドル)→パワー型
→(241回目：相手8ドル／自分52ドル)→バランス型 →243回目で勝利

図2：４班との対戦の時の自分たちの型の移動

**対戦ログから分かったこと①：ディフェンス型の前提の再検討**

- 自分たちは最初の200回、相手に関係なくディフェンス型を採用
- 目的：自分の掛け金を抑えて長期戦に持ち込み、相手の傾向を観察する
- しかし、第2戦では105回目で敗北し、観察期間を待たずに終了
  **⇒「ディフェンス型で安全に情報収集できる」という前提が崩れた**

# コンテスト結果の評価.3

**４班との対戦**

**対戦ログから分かったこと②：自分のカード予測の失敗**

- ディフェンス型では、基本的に自分たちがコールかドロップを選べる立場にある
- しかし、選択肢を活かせていなかった
- 相手の賭け金パターン（※ブラフを除く）
  - ・カード4～8：1ドル
  - ・カード9以上：2ドル
  - ・カード2～4：5ドル
- 自分たちは、「相手が今と同じ金額を過去に賭けたときのカードの中央値と平均値の加重平均」を使って自分のカードを予測していた
- しかし、1～2ドルを賭けてくる範囲が広すぎ、

  **⇒予測値と実際のカードとのギャップが大きくなった**

  **⇒ドロップ／コール判断を誤る場面が多発**

**⇒これが①で分かった 「ディフェンス型で安全に情報収集できるという前提が崩れた」ことの原因**

# コンテスト結果の評価.4

**4班との対戦**

**対戦ログから分かったこと③：分岐のプログラムと戦略の関係性の再検討**

- 1、3戦目はどちらも200回時点で自分たちが優勢だった
- だが、
  - ・1戦目はバランス型に移行 → 敗北
  - ・3戦目はパワー型に移行 → 勝利
- 本来の考え方：
  - ・パワー型 → 相手が1～5ドルをまんべんなく賭けるとき有効
  - ・バランス型 → 相手が1～2ドルを主に賭けるとき有効
- つまり、1～2ドルを主に出す4班にはバランス型を選ぶべきだった
  - → 3戦目では誤ってパワー型を選んだ
  - **⇒型をどう分岐させるか、プログラム自体を見直す必要がある**
- 誤ってパワー型を選んだ時は勝利
  - **⇒「バランス型はディフェンス型に強い」という仮定が揺らいだ**

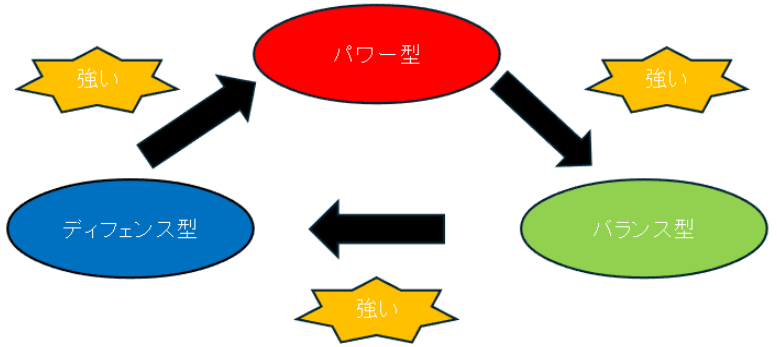


図3：3つの型の関係性（再掲）

# コンテスト結果の評価.5

**5班 (2-1で勝利)**

- Ourcard 2.3.4 TheyBet 5
- Ourcard 13.14 TheBet 1or5
- Ourcard from5to12 TheyBet 1

→ パワー型と判断しディフェンス型で戦っていた

自分のカードを予測する方法が平均値と中央値のみ
→ 5ドルをかけられた際の平均値が 7前後になってしまっていた

→ 1班と同様に自分のカードを予測できず運になっていた
(実際に対戦ログを見ればシーソー状態であった)

# コンテスト全体を踏まえた問題点と改善点.1

**1．型の分岐の失敗**

分岐失敗例：1ドル30%・2ドル40%のとき →実際はディフェンス型だが、条件上は

バランス型に分類される

**⇒ 戦略イメージとプログラムの条件がかみ合わず、誤った型を選択**

**従来の分岐条件（誤った基準）**

- パワー型：1ドルベットが40%を超えているかつ2ドルより5ドルベットのほうが多いとき
- ディフェンス型：相手の1ドルベットが40%を超えているかつ5ドルより2ドルベットのほうが多いとき
- バランス型：1ドルベットが40%に満たないとき

→ 1ドルベットが40%を超えるかどうかを**分岐基準**にしていた

**⇒ 1ドルベットの割合に依存しすぎていた**

# コンテスト全体を踏まえた問題点と改善点.2

**1．型の分岐の失敗（続き）**

**改善案：より実態に沿った分岐基準**

- パワー型：1ドル＋5ドルの割合が解析から得られた一定値以上
- ディフェンス型：1ドル＋2ドルの割合が解析から得られた一定値以上
- バランス型：上記どちらにも当てはまらない場合

  → **賭け金の組み合わせ全体で判断**することで、型選択の精度を高める

# コンテスト全体を踏まえた問題点と改善点.2

**2. 自分のカードの予測の失敗**

失敗例：ログ解析から1班は右下の図のように賭けていた

相手が5ドルを賭けている時に平均と中央値から自分のカードは8前後

相手が1ドルを賭けている時には同様にして自分のカードは8前後

| 自分のカード | 相手のベット |
|---|---|
| 2 or 14 | 5ドル |
| From 3 to 13 | 1ドル |

図4：1班の賭け方

**従来の自分のカードの予測**

「相手が今と同じ金額を過去に賭けたときのカードの
中央値と平均値の加重平均」を使って自分のカードを予測

**⇒ 相手の型に関係なく、一律な予測手法を使ってた**

**改善案**

相手の型ごとに自分のカードの予測する方法を変える

# コンテスト全体の評価

・200回目で 分岐が正しくできた試合 → 10試合(うち5勝5敗)

分岐が正しく行われなかった試合 → 1試合(うち1勝0敗)

200回になる前に終了した試合 → 1試合(うち0勝1敗)

・相手のシステムの予測はとてもうまくいった

→83%(10/12試合)の確率で正しく予測できた

→そのうち50%の確率でしか勝ててない

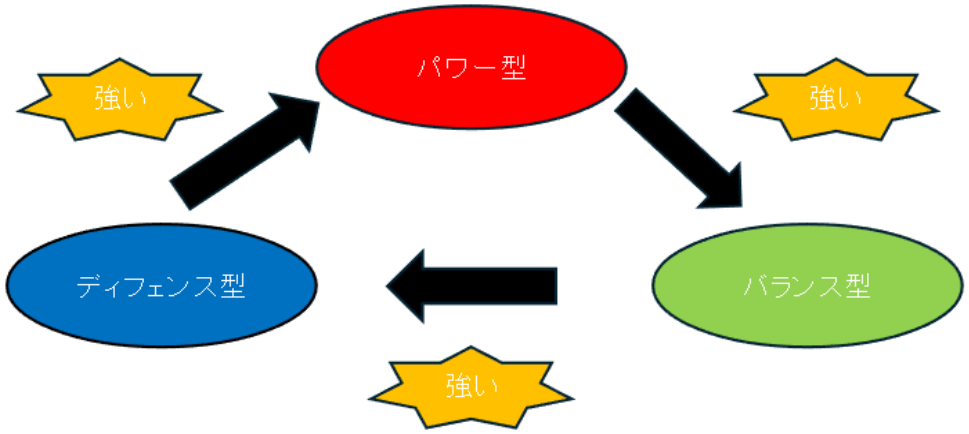


図2:3つの型の関係性
（再掲）

・自分のカードの予測がうまくできていなかった

⇒コールorドロップの判断もうまくいかなかった

三すくみの関係でうまく分岐しても自分のカードの予測のミスで勝率が伸びなかったり、
そもそも三すくみの強さ関係が正しいのかといった部分には疑問が残る結果となる。

# まとめ

- **実験全体から学んだこと**

戦術が三すくみの関係になっていて、一つのプログラムを変えると、今度は別の戦術に弱くなってしまうことがあり、そのバランスを取りながら改良していくのがとても面白かったです。また、チームで取り組むことで、さまざまな視点や考え方に触れることができ、一人では思いつかないアイデアも生まれました。

**▪感想**

黒田にゃんた：プログラムを複雑にしようとするよりも、シンプルにした方がうまくいくこともあると感じました。

白雪えなぴ：結果は残念だったが、自分たちの考えうる強い戦略を実装にまで落とし込むことができてよかった。

氷川ぺんた：個人開発と違いチーム開発では、可読性を高めることやより簡単なロジックを用いることが大事だと気づかされた。

犬塚わん：戦略において前提を立てることの不可欠だが、それが崩れると戦略全体を揺らぐことから、前提を設定することの難しさを痛感した。