

# 競技部門のルール

今年の競技部門では、マス目に区切られたフィールド上で、いかに多くの陣地を占有できるかを競う、陣取りゲームを実施します。試合は1対1の2チームで行われ、各チームは1名の司令塔と、司令塔の指示に沿って行動する2名のエージェントで構成されます。

## ● フィールド

フィールドは最大で縦12×横12＝合計144のマスで構成され、各マスには-16以上16以下の点数が割り当てられます。マスの数は80以上で、試合ごとに設定されます。各マスの点数は、QRコードで司令塔のみに提示されます。1マスの大きさは50cm×50cmで、一方のチームが不利にならないよう、エージェントの配置位置は中央線に対して、必ず水平、垂直、または水平かつ垂直、に線対称に配置されます。

## ● 司令塔エリア

200cm×100cmの広さで、電源は用意されません。司令塔はこのエリアの外に出てはいけません。エリア内に180cm×45cm（高さ70cm）のテーブル1台と椅子1脚が用意されます。指示伝達用の道具としてA4サイズのトランプのみを使うことができます。音声の使用は不可です。

## ● 試合の進行

試合に先立ち、両チームの司令塔に対して各マスの点数とエージェントの配置位置を示すQRコードが見えない状態で渡されます。エージェントには、配置位置のみが提示されます。

試合開始に合わせて、司令塔はQRコードを見ることができます。

試合はターン制で進行し、1ターン内にi. 作戦ステップ、ii. 意思表示ステップ、iii. 行動ステップの3つを行います。作戦ステップでは、エージェントはマスに留まったままで、司令塔がエージェントに指示を送ります。次の意思表示ステップでは、現在のマスに接する8方向のマスに移動する

か、もしくはマスに置いてあるタイルを取り除くか、あるいは何もしないか、を4人のエージェントが手の動きで同時に意思表示します。最後の行動ステップで意思表示の内容を行動に移しますが、例えば同じマスに複数のエージェントが移動を意思表示したような場合は無効となります。移動するとそのマスに自チームのタイルが置かれます。意思表示ステップと次の意思表示ステップの間を1ターンと定義し、試合前に1ターンの時間と試合のターン数が通知されます。与えられたターン数に達したら、試合が終了します。

## ● 得点

タイルを置いたマスの合計点をタイルポイント、タイルが囲んだマスの合計点を領域ポイントとし、その合計ポイントで勝敗が決まります。領域ポイントの場合、マスの負の点数は正の値に変更されます。

## ● 使用可能機器

エージェントは何も用いることができませんが、司令塔はどのような機器も持ち込むことができます。ただし、持ち込み機器と外部との通信は認めません。また、機器はテーブル上もしくはテーブル下の床に設置することができますが、テーブルのサイズからはみ出して設置することはできません。

## ● 試合形式

1対1のフィールドを最大12試合同時に行います。予選はリーグ戦、敗者復活はリンクリーグ戦、決勝はトーナメントで実施します。

## ● 勝敗決定方法

勝敗判定は以下の優先順位で決定します。

1. 合計ポイント
2. 合計ポイントが同じときはタイルポイント
3. 合計ポイントもタイルポイントも等しいときはサイコロの目

## 競技部門の組合せ（１）

### 予選リーグ戦の組合せ

| リーグ | 記号   | チーム |
|-----|------|-----|
| P1  | P1-a | 米子  |
|     | P1-b | 長野  |
|     | P1-c | 岐阜  |
|     | P1-d | 新居浜 |

| リーグ | 記号   | チーム    |
|-----|------|--------|
| P2  | P2-a | 福島     |
|     | P2-b | 神戸市立   |
|     | P2-c | 久留米    |
|     | P2-d | 香川（詫間） |

| リーグ | 記号   | チーム  |
|-----|------|------|
| P3  | P3-a | 釧路   |
|     | P3-b | 小山   |
|     | P3-c | 弓削商船 |
|     | P3-d | 旭川   |

| リーグ | 記号   | チーム    |
|-----|------|--------|
| P4  | P4-a | 群馬     |
|     | P4-b | 仙台（名取） |
|     | P4-c | 熊本（八代） |
|     | P4-d | 熊本（熊本） |

| リーグ | 記号   | チーム |
|-----|------|-----|
| P5  | P5-a | 奈良  |
|     | P5-b | 茨城  |
|     | P5-c | 函館  |
|     | P5-d | 長岡  |

| リーグ | 記号   | チーム    |
|-----|------|--------|
| P6  | P6-a | 明石     |
|     | P6-b | 高知     |
|     | P6-c | 鳥羽商船   |
|     | P6-d | 都立（荒川） |

| リーグ | 記号   | チーム    |
|-----|------|--------|
| P7  | P7-a | 香川（高松） |
|     | P7-b | 宇部     |
|     | P7-c | 呉      |
|     | P7-d | 豊田     |

| リーグ | 記号   | チーム    |
|-----|------|--------|
| P8  | P8-a | 大分     |
|     | P8-b | 一関     |
|     | P8-c | 都立（品川） |
|     | P8-d | 広島商船   |

| リーグ | 記号   | チーム  |
|-----|------|------|
| P9  | P9-a | 松江   |
|     | P9-b | 大阪府大 |
|     | P9-c | 徳山   |
|     | P9-d | 鶴岡   |

| リーグ | 記号    | チーム |
|-----|-------|-----|
| P10 | P10-a | 木更津 |
|     | P10-b | 沼津  |
|     | P10-c | 沖縄  |
|     | P10-d | 舞鶴  |

| リーグ | 記号    | チーム  |
|-----|-------|------|
| P11 | P11-a | 苫小牧  |
|     | P11-b | 秋田   |
|     | P11-c | 大島商船 |
|     | P11-d | 鈴鹿   |

| リーグ | 記号    | チーム        |
|-----|-------|------------|
| P12 | P12-a | 有明         |
|     | P12-b | 国際         |
|     | P12-c | 東京         |
|     | P12-d | モンゴル科学技術大学 |

| リーグ | 記号    | チーム     |
|-----|-------|---------|
| P13 | P13-a | 津山      |
|     | P13-b | 八戸      |
|     | P13-c | 北九州     |
|     | P13-d | ハノイ国家大学 |

| リーグ | 記号    | チーム       |
|-----|-------|-----------|
| P14 | P14-a | 佐世保       |
|     | P14-b | 仙台（広瀬）    |
|     | P14-c | 石川        |
|     | P14-d | ペトロナス工科大学 |

| リーグ | 記号    | チーム    |
|-----|-------|--------|
| P15 | P15-a | 富山（射水） |
|     | P15-b | 鹿児島    |
|     | P15-c | 福井     |
|     | P15-d | 香港 VTC |

| リーグ | 記号    | チーム  |
|-----|-------|------|
| P16 | P16-a | 都城   |
|     | P16-b | 阿南   |
|     | P16-c | サレジオ |

※各リーグ 1 位のチームが決勝トーナメントに進出します。

※勝敗が同じ場合は合計ポイント数が多いチームを上位とします。合計ポイントが等しい場合、タイルポイントが大きいチームを上位とします。合計ポイントもタイルポイントも等しい場合は、サイコロの数が大きい方を上位とします。

※予選リーグには、海外チームがオープン参加します。海外チームの成績は、国内チームの順位に関係ありません。

## 競技部門の組合せ (2)

予選リーグ戦の対戦表

第 1 試合

| フィールド | 記号      |
|-------|---------|
| A     | P 1 - a |
|       | P 1 - b |
| B     | P 1 - c |
|       | P 1 - d |
| C     | P 2 - a |
|       | P 2 - b |
| D     | P 2 - c |
|       | P 2 - d |
| E     | P 3 - a |
|       | P 3 - b |
| F     | P 3 - c |
|       | P 3 - d |
| G     | P 4 - a |
|       | P 4 - b |
| H     | P 4 - c |
|       | P 4 - d |
| I     | P 5 - a |
|       | P 5 - b |
| J     | P 5 - c |
|       | P 5 - d |
| K     | P 6 - a |
|       | P 6 - b |
| L     | P 6 - c |
|       | P 6 - d |

第 2 試合

| フィールド | 記号      |
|-------|---------|
| A     | P 7 - a |
|       | P 7 - b |
| B     | P 7 - c |
|       | P 7 - d |
| C     | P 8 - a |
|       | P 8 - b |
| D     | P 8 - c |
|       | P 8 - d |
| E     | P 9 - a |
|       | P 9 - b |
| F     | P 9 - c |
|       | P 9 - d |
| G     | P10-a   |
|       | P10-b   |
| H     | P10-c   |
|       | P10-d   |
| I     | P11-a   |
|       | P11-b   |
| J     | P11-c   |
|       | P11-d   |
| K     | P12-a   |
|       | P12-b   |
| L     | P12-c   |
|       | P12-d   |

第 3 試合

| フィールド | 記号      |
|-------|---------|
| A     | P13-a   |
|       | P13-b   |
| B     | P13-c   |
|       | P13-d   |
| C     | P14-a   |
|       | P14-b   |
| D     | P14-c   |
|       | P14-d   |
| E     | P15-a   |
|       | P15-b   |
| F     | P15-c   |
|       | P15-d   |
| G     | P16-a   |
|       | P16-b   |
| H     |         |
|       |         |
| I     | P 1 - a |
|       | P 1 - c |
| J     | P 1 - b |
|       | P 1 - d |
| K     | P 2 - a |
|       | P 2 - c |
| L     | P 2 - b |
|       | P 2 - d |

第 4 試合

| フィールド | 記号      |
|-------|---------|
| A     | P 3 - a |
|       | P 3 - c |
| B     | P 3 - b |
|       | P 3 - d |
| C     | P 4 - a |
|       | P 4 - c |
| D     | P 4 - b |
|       | P 4 - d |
| E     | P 5 - a |
|       | P 5 - c |
| F     | P 5 - b |
|       | P 5 - d |
| G     | P 6 - a |
|       | P 6 - c |
| H     | P 6 - b |
|       | P 6 - d |
| I     | P 7 - a |
|       | P 7 - c |
| J     | P 7 - b |
|       | P 7 - d |
| K     | P 8 - a |
|       | P 8 - c |
| L     | P 8 - b |
|       | P 8 - d |

第 5 試合

| フィールド | 記号      |
|-------|---------|
| A     | P 9 - a |
|       | P 9 - c |
| B     | P 9 - b |
|       | P 9 - d |
| C     | P10-a   |
|       | P10-c   |
| D     | P10-b   |
|       | P10-d   |
| E     | P11-a   |
|       | P11-c   |
| F     | P11-b   |
|       | P11-d   |
| G     | P12-a   |
|       | P12-c   |
| H     | P12-b   |
|       | P12-d   |
| I     | P13-a   |
|       | P13-c   |
| J     | P13-b   |
|       | P13-d   |
| K     | P14-a   |
|       | P14-c   |
| L     | P14-b   |
|       | P14-d   |

第 6 試合

| フィールド | 記号      |
|-------|---------|
| A     | P15-a   |
|       | P15-c   |
| B     | P15-b   |
|       | P15-d   |
| C     | P16-b   |
|       | P16-c   |
| D     |         |
|       |         |
| E     | P 1 - d |
|       | P 1 - a |
| F     | P 1 - c |
|       | P 1 - b |
| G     | P 2 - d |
|       | P 2 - a |
| H     | P 2 - c |
|       | P 2 - b |
| I     | P 3 - d |
|       | P 3 - a |
| J     | P 3 - c |
|       | P 3 - b |
| K     | P 4 - d |
|       | P 4 - a |
| L     | P 4 - c |
|       | P 4 - b |

第 7 試合

| フィールド | 記号      |
|-------|---------|
| A     | P 5 - d |
|       | P 5 - a |
| B     | P 5 - c |
|       | P 5 - b |
| C     | P 6 - d |
|       | P 6 - a |
| D     | P 6 - c |
|       | P 6 - b |
| E     | P 7 - d |
|       | P 7 - a |
| F     | P 7 - c |
|       | P 7 - b |
| G     | P 8 - d |
|       | P 8 - a |
| H     | P 8 - c |
|       | P 8 - b |
| I     | P 9 - d |
|       | P 9 - a |
| J     | P 9 - c |
|       | P 9 - b |
| K     | P10-d   |
|       | P10-a   |
| L     | P10-c   |
|       | P10-b   |

第 8 試合

| フィールド | 記号    |
|-------|-------|
| A     | P11-d |
|       | P11-a |
| B     | P11-c |
|       | P11-b |
| C     | P12-d |
|       | P12-a |
| D     | P12-c |
|       | P12-b |
| E     | P13-d |
|       | P13-a |
| F     | P13-c |
|       | P13-b |
| G     | P14-d |
|       | P14-a |
| H     | P14-c |
|       | P14-b |
| I     | P15-d |
|       | P15-a |
| J     | P15-c |
|       | P15-b |
| K     | P16-c |
|       | P16-a |
| L     |       |
|       |       |

※対戦表中の上段のチームがステージに向かって左側の司令塔を使います。

## 競技部門の組合せ (3)

### 敗者復活リーグ戦の組合せ

| リーグ | 記号   | チーム   |
|-----|------|-------|
| L1  | L1-a | P1-2  |
|     | L1-b | P9-2  |
|     | L1-c | P8-3  |
|     | L1-d | P16-3 |
|     | L1-e | P4-4  |

| リーグ | 記号   | チーム   |
|-----|------|-------|
| L2  | L2-a | P2-2  |
|     | L2-b | P10-2 |
|     | L2-c | P1-3  |
|     | L2-d | P9-3  |
|     | L2-e | P5-4  |

| リーグ | 記号   | チーム   |
|-----|------|-------|
| L3  | L3-a | P3-2  |
|     | L3-b | P11-2 |
|     | L3-c | P2-3  |
|     | L3-d | P10-3 |
|     | L3-e | P6-4  |

| リーグ | 記号   | チーム   |
|-----|------|-------|
| L4  | L4-a | P4-2  |
|     | L4-b | P12-2 |
|     | L4-c | P3-3  |
|     | L4-d | P11-3 |
|     | L4-e | P7-4  |

| リーグ | 記号   | チーム   |
|-----|------|-------|
| L5  | L5-a | P5-2  |
|     | L5-b | P13-2 |
|     | L5-c | P4-3  |
|     | L5-d | P12-3 |
|     | L5-e | P8-4  |

| リーグ | 記号   | チーム   |
|-----|------|-------|
| L6  | L6-a | P6-2  |
|     | L6-b | P14-2 |
|     | L6-c | P5-3  |
|     | L6-d | P13-3 |
|     | L6-e | P1-4  |
|     | L6-f | P9-4  |

| リーグ | 記号   | チーム   |
|-----|------|-------|
| L7  | L7-a | P7-2  |
|     | L7-b | P15-2 |
|     | L7-c | P6-3  |
|     | L7-d | P14-3 |
|     | L7-e | P2-4  |
|     | L7-f | P10-4 |

| リーグ | 記号   | チーム   |
|-----|------|-------|
| L8  | L8-a | P8-2  |
|     | L8-b | P16-2 |
|     | L8-c | P7-3  |
|     | L8-d | P15-3 |
|     | L8-e | P3-4  |
|     | L8-f | P11-4 |

### 敗者復活リーグ戦の対戦表

#### 第1試合

| フィールド | 記号   |
|-------|------|
| A     | L1-a |
|       | L1-b |
| B     | L1-c |
|       | L1-d |
| C     | L2-a |
|       | L2-b |
| D     | L2-c |
|       | L2-d |
| E     | L3-a |
|       | L3-b |
| F     | L3-c |
|       | L3-d |
| G     | L4-a |
|       | L4-b |
| H     | L4-c |
|       | L4-d |
| I     | L5-a |
|       | L5-b |
| J     | L5-c |
|       | L5-d |
| K     | L6-a |
|       | L6-b |
| L     | L6-c |
|       | L6-d |

#### 第2試合

| フィールド | 記号   |
|-------|------|
| A     | L1-a |
|       | L1-e |
| B     | L6-e |
|       | L6-f |
| C     | L2-a |
|       | L2-e |
| D     | L7-a |
|       | L7-b |
| E     | L3-a |
|       | L3-e |
| F     | L7-c |
|       | L7-d |
| G     | L4-a |
|       | L4-e |
| H     | L7-e |
|       | L7-f |
| I     | L5-a |
|       | L5-e |
| J     | L8-a |
|       | L8-b |
| K     | L8-c |
|       | L8-d |
| L     | L8-e |
|       | L8-f |

#### 第3試合

| フィールド | 記号   |
|-------|------|
| A     | L1-d |
|       | L1-e |
| B     | L1-b |
|       | L1-c |
| C     | L2-d |
|       | L2-e |
| D     | L2-b |
|       | L2-c |
| E     | L3-d |
|       | L3-e |
| F     | L3-b |
|       | L3-c |
| G     | L4-d |
|       | L4-e |
| H     | L4-b |
|       | L4-c |
| I     | L5-d |
|       | L5-e |
| J     | L5-b |
|       | L5-c |
| K     | L6-b |
|       | L6-c |
| L     | L6-d |
|       | L6-e |

#### 第4試合

| フィールド | 記号   |
|-------|------|
| A     | L6-f |
|       | L6-a |
| B     | L7-b |
|       | L7-c |
| C     | L7-d |
|       | L7-e |
| D     | L7-f |
|       | L7-a |
| E     | L8-b |
|       | L8-c |
| F     | L8-d |
|       | L8-e |
| G     | L8-f |
|       | L8-a |
| H     |      |
|       |      |
| I     |      |
|       |      |
| J     |      |
|       |      |
| K     |      |
|       |      |
| L     |      |
|       |      |

※チーム X-Y とは予選 X リーグ、第 Y 位のチームを表します。

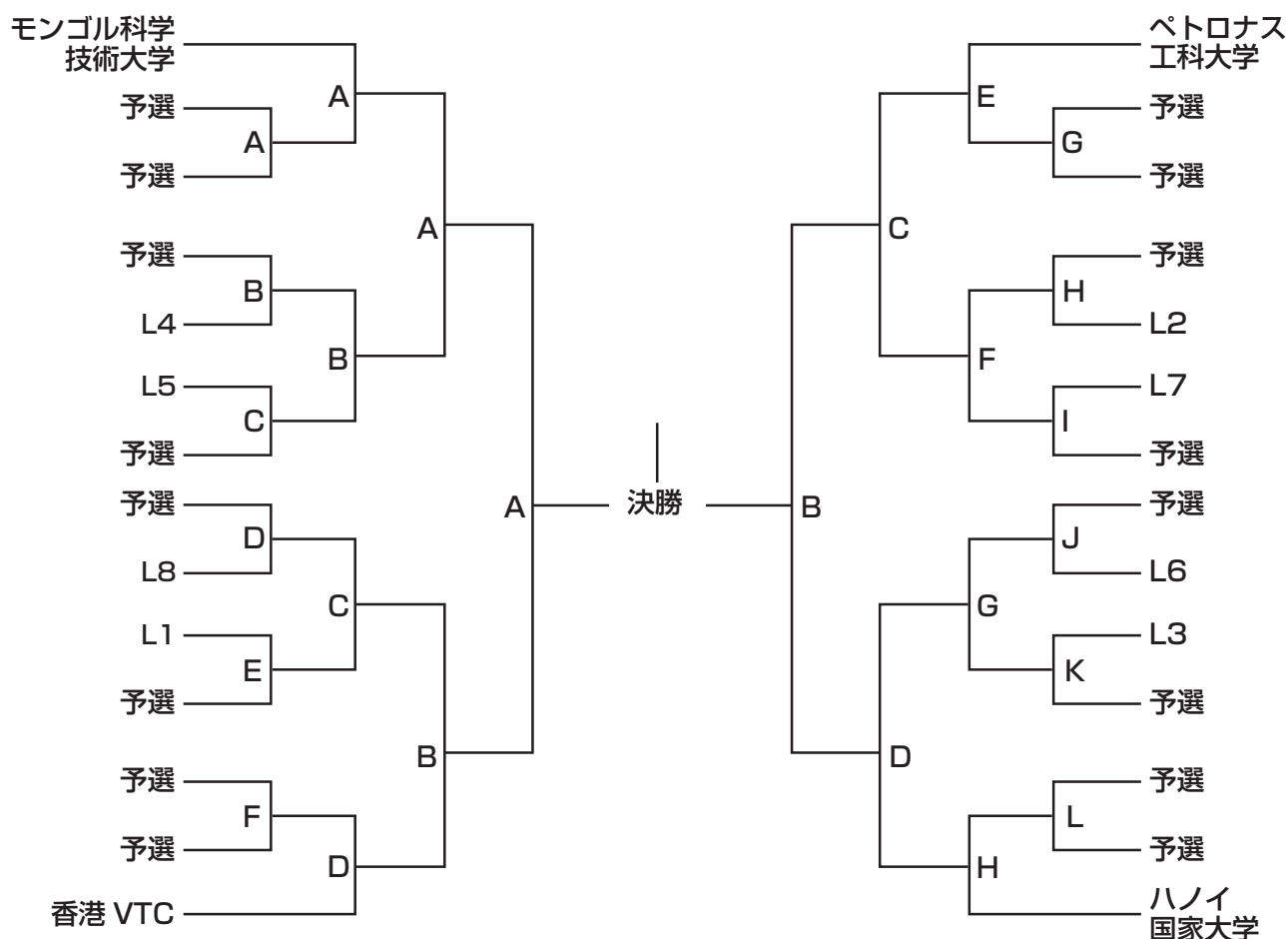
※敗者復活は 5、または 6 チームのリンク・リーグ戦で行い、各リーグ 1 位のチームが決勝トーナメントに進出します。

※勝敗が同じ場合は、予選リーグと同様に合計ポイント、タイルポイント、サイコロによって順位を決めます。

※対戦表中の上段のチームがステージに向かって左側の司令塔を使います。

## 競技部門の組合せ（4）

### 決勝トーナメントの組合せ



※「予選」には、予選リーグ 1 位のチームが入ります。いずれのチームが入るかは、予選リーグ終了後に抽選によって決めます。

※ L1 から L8 は、敗者復活リーグ L1 から L8 にて 1 位になったチームです。

※トーナメントの左側（図の左側では上、右側では下）のチームがステージに向かって左側の司令塔を使います。

※図の A から L はフィールドを表します。1 回戦であれば A から L までの 12 試合が同時に実施されます。2 回戦も A から H までの 8 試合が同時に行われます。準々決勝・準決勝・決勝も同様です。

※ 3 位決定戦は決勝戦と並行して行われます。

※ 2 回戦以降は NAPROCK 国際プロコン（国際大会）を兼ねて実施されます。国内チーム・海外チームの区別なく、試合の成績によって国際大会での順位が決まります。

※全国高専プログラミングコンテストの順位は、海外チームを除きます。

## 1 レポートの柵

秋 田

米田 凌（4年） 鈴木 優也（4年）  
戸賀瀬 駿（4年） 竹下 大樹（教員）

### 1. はじめに

どのように移動すれば効率良くポイントがとれるのかを考えるのは、人力では時間がかかるため困難である。そこで、重み付け評価のアルゴリズムを用いて最適ルートの解析をし、ゲームを進行していく。

### 2. データの入力

盤面の情報はQRコードとなっているため、QRコードを読み取り、データ化させる。

### 3. 伝達方法

エージェントへの指示は、相手に解読されづらいもの、自分達がすぐに解読できる方法にする。今回は、2枚のトランプを使い、マークや数字の組み合わせで方向を指定する。

### 4. 重み付け評価

#### 4.1 仕様と目的

計算を用いて、どこへ移動するのがいいのかを解析する。考える時間は制限があるため、この評価方法を用いて、最適ルートの解析をしつつ、考える時間の短縮を目指す。

#### 4.2 アルゴリズム

盤面の点数、自分と相手のエージェントの配置、次に動くことのできるマスなどの情報をもとに、評価項目にそれぞれ重みをつけ、最も重みの大きいマスを求め、次の行動を決める。

### 5. 開発環境

開発言語：C, C++

使用IDE：Visual Studio

## 2 人カタイムプロコン

明 石

大谷 直輝（2年） 岡田 颯太（2年）  
東 志拓（1年） 佐村 敏治（教員）

### 1. システム概要

配布されたQRコードをカメラで読み取り、エージェントの初期位置や、フィールドの状況を把握する。毎ターン敵エージェントの位置を入力し、次のターンの行動を見やすい形で出力する。

### 2. 戦略

負のマスは数が少ないうえに囲うリスクが高い（途切れさせられる）と考えられるので、基本的には正のマスを優先してせめる。

### 3. 司令塔の指示決定アルゴリズム

#### 3.1 ビームサーチ法

その時点での最良手だけでなく、複数の状態をもつて探索することで多様性を持たせる。

### 3.2 機械学習

事前に自分たちで競技をした結果を学習させ、それを利用して最良手を探す。

### 4. エージェントへの指示伝達方法

エージェントへの指示にはトランプを使うことになるが、秘匿性よりも伝達の容易さを重視する。また、ボディランゲージも場合によっては利用して、確実な情報伝達をおこなう。

### 5. 開発環境

OS：Windows10, Ubuntu18

言語：Java, C++, Python

ライブラリ：ZXing, TensorFlow

## 3 シンタローズ

鈴 鹿

松井信太郎（5年）中西 虹太（4年）  
高橋 祐策（5年）青山 俊弘（教員）

### 1. はじめに

今回の競技は、マス目に区切られたフィールドでどれだけ多くの陣地を占領できるかを競うものである。処理の流れとしては、QRコードからフィールド情報の読み取り、競技中の味方・相手のエージェントの位置取得、陣地を占領していくアルゴリズム、そして処理結果を出力である。

### 2. 入力1:フィールドの取得

競技開始直後にQRコードが配られる。Webカメラを用いてQRコードを読み取る。そのデータを元にフィールドを作成する。

### 3. 入力2:エージェントの位置の取得

カメラからフィールドを読み取り、画像処理を用いて位置情報を得て、適宜更新していく。画像処理が不十分な場合は、手入力で補えるような仕様である。

### 4. 処理:アルゴリズム

強化学習を用いてAIを作成し、当日は学習済みのAIを使用する。学習では、Q学習やモンテカルロ法など様々なアルゴリズムを用い、最も良いスコアを出したAIを採用する。

### 5. 出力:UIについて

競技途中、フィールド内のエージェントに指示を出せるようなUIを用いる。また、3でも記述した通り、競技途中の相手エージェントの入力等も行えるような仕様である。

### 6. 開発環境について

言語: python , go

ライブラリ: OpenCV , OpenAI gym

## 4 あ ^ ~よく寝た !!

香 川  
(詫間)

人見 俊（3年）三瀬リキト（2年）  
漆原 和輝（2年）宮崎 貴大（教員）

### 1. はじめに

1 台の PC と 1 台の Android 端末を用いる。Android 端末で QR コードを読み取り、そのデータを PC に転送し画面に表示する。

### 2. データの取得

QR コードを読み取るために専用のアプリケーションを開発し、PC で読み取りやすいデータ書式に変形して転送する。

### 3. 司令塔の指示決定アルゴリズム

タイルポイントとエリアポイントのどちらも考慮し、最もポイントが高くなるエージェントの移動ルートを、導出するアルゴリズムを作成すればよい。しかし、そのようなアルゴリズムを作成することは、非常に困難であるという

判断を下さざるを得なかった。そこでポイントを複合的に判断し解を導き出せる、人を用いることにした。

### 4. エージェントへの指示伝達方法

エージェントへは野球部などでよく用いられる指示伝達方法を使う。

### 5. 開発環境

OS Windows10, Android7.0

使用言語 C/C++, Java

使用ソフト Microsoft Visual Studio 2017 Community  
Android Studio 2.3

使用ライブラリ OpenCV, Boost, Eigen

## 5 高専の陣取り概論

長 野

岩瀬 天真（3年）上松崇太郎（3年）  
高野 徳次（2年）伊藤 祥一（教員）

### 1. 概要

最初に、配布される QR コードからフィールド情報を読み取る。ターンごとに2人のエージェントの移動先の候補を表示し、それに基づいてエージェントに指示をする。意図した通りに移動できない可能性があるため、移動が完了した後で両チームのエージェントの実際の移動位置を入力し、それに基づいて次のターンの計算をする。

### 2. 解法

#### 2.1 移動方向の決定方法

エージェントが移動したときのフィールドの状態に評価値を与え、評価値が最も高い移動方向を最終的な決定とする。この計算は、エージェントが移動可能な全方向について行う。

#### 2.2 評価関数

評価値を計算する評価関数は、深層学習を用いて作成す

る。まず、ネットワークの出力値は0から1の値をとる。事前に多数の対戦データを生成し、対戦データの実際の勝者と、学習したネットワークから推論した勝者を比較して、出力が1に近いほど勝利する確率が高くなるようにネットワークを学習させる。学習させたネットワークで評価値を推論し、この出力値を用いて評価関数を作成する。

### 3. エージェントへの指示方法

カードの色と番号、ジェスチャーを組み合わせ、指示を秘匿することよりも、エージェントに速く確実に伝達することを優先する。

### 4. 開発環境

OS: Windows10

言語: C/C++, Python

ライブラリ・フレームワーク: Siv3D, PyTorch

IDE: Visual Studio 2017

## 6 囲碁アルゴ

鳥羽商船

田中 潤一（2年）堀口 凱生（3年）  
弓削 隼大（1年）土田 隼之（教員）

### 1. はじめに

今回の競技では、マス目に区切られたフィールド上で、いかに多くの陣地を占有できるかを競う陣取りゲームである。

### 2. 入力方法

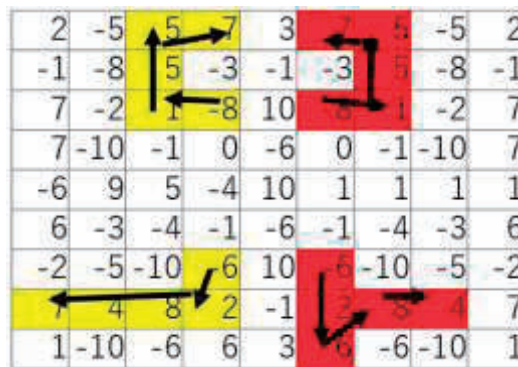
QR コードからフィールド情報を読み込み、csv ファイルで保存して、プログラムへ入力する。

### 3. 手順の算出アルゴリズム

プレイヤーマスを含む複数マスで、得点が大きくなるマス領域を貪欲法で算出する。自分が現在居るマスから、出来るだけ相手の反対方向かつ最大得点となるマスを選択して移動する。これを、複数回繰り返す。なお、相手との距離が一定以上離れている場合には相手が居る側への移動も行う。

右図では、左上の-8点マス、右上の-8点マス、左下の

6点マス、右下の6点マスから手順の算出アルゴリズムを開始した場合の順路を示している。左上の-8点マスからの移動手順の算出を考える。最初の手順では、相手との距離が近い反対方向の1点マスへ移動する。その後は、相手との距離が一定以上離れており、反対方向以外への移動も考慮する。そして、それぞれの地点での最大得点マスの5点マス、5点マス、7点マスへの移動手順を出力する。



## 7 クレマスカ？

呉

今村 圭 (5 年) 水本 直希 (3 年)  
宮本 大哉 (2 年) 藤井 敏則 (教員)

### 1. はじめに

今回の競技では、ディープラーニングを利用し最善手を求める方法が最も適していると考えました。そこで方策ネットワークを構築し、次の手を推測するシステムを開発します。

### 2. ステップ毎の進行

#### (1) 作戦ステップ

作戦ステップでは、推論を実行する機能と推論の結果を表示する機能を主に提供します。

#### (2) 意思表示ステップ

意思表示ステップでは、推論の結果、司令塔がどのような行動を取ればよいかを GUI で表示します。

#### (3) 行動ステップ

行動ステップでは、実際にエージェントがどのように動いたかを入力します。

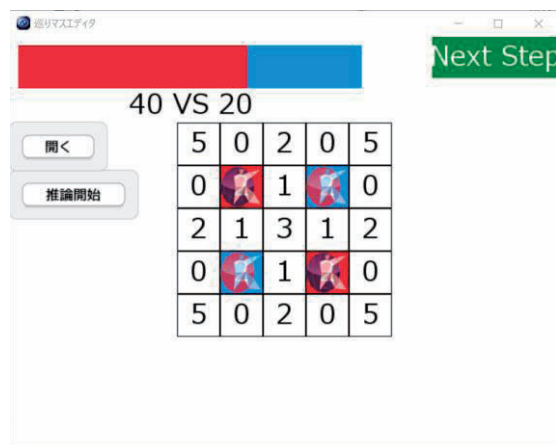


図 GUI のイメージ

### 3. 環境

言語 : C++, Python

ライブラリ/フレームワーク : Siv3D, Chainer など

## 8 とらんポータル

新居 浜

杉光 拓実 (4 年) 松本 楓真 (3 年)  
原 章也 (3 年) 占部 弘治 (教員)

### 1. はじめに

今回の競技は、占有した陣地のポイントによって勝敗が決まる陣取りゲームである。したがって、自チームのポイントの最大化を狙った正確な指示が司令塔から出せる設計を考える。

### 2. アルゴリズム

1 ターンの短い間に探索して最適解を求め、その結果から司令塔が指示を出すのは難しいと考えた。

そこで、フィールド上から得られる様々なデータに対して予め用意しておいた係数を掛けたものの総和と司令塔の判断を数値化したものの合計が最大となるものを最適解とする。これによって計算を単純化することができ、毎ターンその結果から司令塔が指示を出せると考える。

### 3. 指示の伝達について

基本的にはエージェント 1 人につきトランプ 2 枚で指示を出す。エージェントが正しい行動でない動きをした場合のみボディランゲージによる指示を行う。

### 4. UI について

司令塔は 1 ターンの間、『動きの入力』、『トランプまたはボディランゲージによる指示』を行わなければならない。そこで、指示に最大限の時間を掛けれるように GUI を用意し、指示によるミスがないようにする。

### 5. 開発環境

言語: C, C++, C#, python

環境: Visual Studio 2015/2017

OS: Windows7/8.1/10

## 1. システム概要

事前に用意した複数のアルゴリズムを一台の PC 上で実行、および試合の経過をシミュレートする。その実行結果から最善のものを選択し、人の手も加えながら各エージェントの進行方向を決定していく。

## 2. エージェントの移動

## 2.1 アルゴリズム

事前に設定したパラメータを元に数手先までパターンを計算し、最も結果が良かった手を選んでいく。また、序盤・中盤・終盤でタイルポイントと領域ポイントの優先度を調整しながら計算させる。

## 2.2 GUI

自チームを赤、相手チームを青と見なして、解析した情報を分かりやすいように視覚化する。またタイルの点数が正か負か、囲まれている領域はどこかなどといった情報も

オプションで表示できるように実装する。

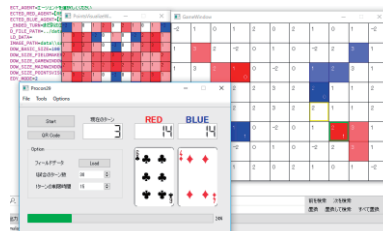


図 GUI の例

## 3. エージェントへの伝達

PC 上でのプログラム実行結果と司令塔の判断によって決定された進行方向を、事前に決めたルールを元に紐づけされた特定のトランプをコンピューター上に表示する。司令塔はその画面をみて素早くトランプを掲げる。

## 4. 開発環境

言語 : C/C++ IDE : Qt Creator 4.6.1 (Community)

ライブラリ : Qt/OpenCV/QZXing/OpenMP/ZBar

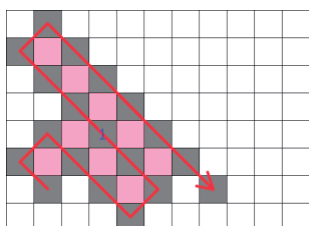
## 1. はじめに

今回の競技は、マス目に区切られたフィールド上で、いかに多くの陣地を占有できるかを競う 陣取りゲームである。競技に挑むに当たって作成するシステムは QR コードからフィールドの情報を取得する部分や適切な移動先を求めるアルゴリズムの処理部分、またその結果を画面上に表示する処理部分からなる。

## 2. アルゴリズム

## 2.1 競技序盤の移動

競技序盤では、規則的に移動する。図 (赤は移動経路、桃色は領域ポイントとなるタイル) のように蛇腹のように折り返すことによってタイルと領域が網目のように並ぶので、タイルが 1 つ取られても (図では 1 と書いたタイル) 辺周辺以外であれば損害はタイルポイント分のみで済む場合が多い。



## 2.2 ミニマックス法

敵味方の行動をゲーム木で考え、市松模様状に動くことを前提としてタイルポイントと領域ポイントの総和が味方は多く、敵は少なくなる最善の手を打つようにする。

## 2.3 競技終盤の移動

終盤は各所に空いた市松模様の穴を埋める事は効率が悪いので、タイルポイントを稼ぐことも交えながら移動するように行動パターンをマクシミン戦略に変化させる。

## 3. 入出力について

## 3.1 入力

フィールドのデータ取得に QR コードが配布されるのでこの情報を入力とする。QR コードは PC の Web カメラを用いて取り込み、データ化する。

## 3.2 出力 (UI)

今回の競技ではフィールドの状況を毎ターン知る必要があるため、人間にとってわかりやすい GUI を表示し、司令塔はシステムが提案したエージェントの動作や移動先に沿ってエージェントに司令を送る。

## 4. 開発環境

言語 : C, C++

エディタ : Visual Studio Code, Emacs

ライブラリ : DX ライブラリ

## 1. はじめに

今回の問題は、自チームだけでなく、他チームの情報を得ることが出来る。限られたターン時間などの中で、よりよい結果を出すために、本チームは複数の解法を採用した。

## 2. 解法アルゴリズム

- ① Beam Search
- ② モンテカルロ法

本チームは主に上記の 2 つのアルゴリズムを採用する。

Beam Search は幅優先探索によって得た盤面に対し、評価関数によって得たスコア順にソートして、上位 N 個の結果を次の探索に用いるアルゴリズムのため、時間制約の厳しい本問題に対して一定の有効性があると考えた。

モンテカルロ法による機械学習は、メイン解法である Beam Search の補助として取り入れる。

## 3. エージェントへの指示伝達

運営側から提供されるトランプを用いると、指示に時間が掛かってしまうため、ハンドサインによってエージェントに指示を行う。単純なサインの場合、対戦相手に指示が予測されてしまうため、複数回に分けてサインを出す。

## 4. 解法のヴィジュアルイズ

司令塔がエージェントに指示をしやすいように、3 で述べたハンドサインの画像を GUI アプリケーション上に表示することで指示にかかる時間を削減する。

## 5. 開発環境・動作環境

[OS] Mac OS High Sierra, Ubuntu 18.04

[Language] D, C++, Ruby, Python3

[IDE・Editor] Neovim, CLion, VisualStudioCode

[Library] OpenCV, ZBar, OpenSiv3D, Tensorflow

## 1. システム概要

私たちのシステムは大きく分けて 3 つの評価基準を設ける。合計スコア、各エージェントの動き、数手先における移動可能マスによる情報によって評価を付ける。これらの評価を用い、総合的に最も優れた手を選択する。

また、システムが明らかに致命的な選択をした場合、人為的に修正することが可能である。

## 2. システムの特徴

1 ターンごとの状況の更新において自チームのエージェントの移動先であるマスの情報をデータで受け渡すことでデータ入力による手間を省く。

アルゴリズムによって致命的な選択がなされた場合には、人の手によって修正が可能になっており、誤作動による影響を抑制出来る。これは、あくまでも戦略に柔軟性を持たせ、より最適な解を導くための手法である。

## 3. システムの流れ

ユーザからの入力を受け、それに応じて評価を行い、最善手を決定する。

ユーザが GUI を用いてエージェントの位置などの情報を入力する。入力された情報に基づき、ある評価基準によって、最善と思われる手を決定する。具体的には、合計スコア、相手エージェントの動き、数手先における移動可能マス数などを加味し、それぞれに重みをつけて判断する。また、終盤の数ターンでは、総当たりによる最善手の探索を行う。

## 4. 開発環境

IDE : VisualStudio 2017

ワークフレーム : Siv3D(August 2016 v2)

言語 : C++ , C

その他 : ZBar(ver0.10) , Github

## 13 for (i=0;j<n;i++)

松 江

鈴木 豪 (3年) 青木 蓮樹 (3年)  
居田 陸人 (2年) 橋下 剛 (教員)

### 1. システム概要

次の手を考える思考プログラムと、ユーザと情報のやり取りをするインターフェースプログラムを作成した。

### 2. 思考プログラム

#### 2.1 基本的な探索方法

自分と相手が交互に動くゲームとみなし、ミニマックス法 (深さ優先探索) を用いて、指し手を決定する。αβ枝刈りなどによって計算量を減らしている。

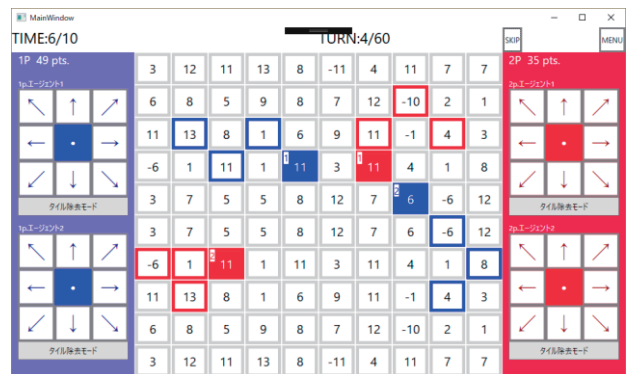
#### 2.2 評価関数

最終的な得点だけでなく、その盤面にたどり着くまでの手数、形なども考慮して評価関数を作った。

### 3. インターフェースプログラム

2つの機能がある。1つは思考プログラムの結果を表示

する機能、もう1つは思考プログラムの動作や盤面の情報を変更する機能である。これにより想定外の事態が起きてもユーザ操作で対処できるようにしている。



### 4. 開発環境

言語 : C#, XAML

IDE : Visual Studio 2017

OS : Windows 10

## 14 ハッピーショーセンライフ

大島商船

河野 冬紀 (3年) 岡村 一矢 (2年)  
常森 俊成 (2年) 北風 裕教 (教員)

### 1. システム概要

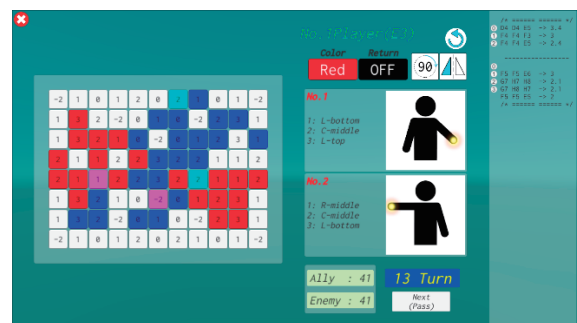
陣取りゲームで計算処理時間も考慮して効率よく得点を重ねるために、ルートの探索範囲を限定したアルゴリズムを提案してシステム開発を試みた。具体的には現在地のマスに近いものほど高い重み値を加えて3マス先までの最適なルート探索を行う手法としている。

### 2. 読み取り・入出力

与えられたQRコードから全要素を“スペース”と“:”で区切って格納し、マスの大きさや得点、エージェントの位置情報をそれぞれ取得した。エージェントの位置に対して、次に進むマスをクリックして進むように工夫を行った。また、誤り操作を訂正する操作や、直接マスの色を編集する操作、全体を回転・反転する操作などを感覚的に操作しやすいGUIを作成した。現在のターン数、マスの得点や色、敵と味方の点数、エージェントの位置、現在入力しているエージェントの名前、ルートの候補、ジェスチャーのポーズなどの機能を充実させて操作性を向上させた。

### 3. アルゴリズム

各マスに与えられた得点情報と取得した陣地情報を参考に、次に取得する陣地の評価を行うアルゴリズムを提案した。評価



で利用する評価値として、重み情報を新たに付加することで、処理量を削減することができるようにした。具体的には、エージェント位置に対して1マス先の得点を×1.0倍とし、2マス先の得点×0.7倍、3マス先の得点×0.4倍で各々算出し、総和を求めることで評価値とした。それを順序付けし、3番目に高い得点までソートすることで、候補となるマスを選定した。これらのユークリッド距離に応じた評価値を求めることで効率的に取得陣地の獲得が可能になった。

### 4. 開発環境・言語

開発環境 : Unity2018.2.0、VisualStudio2017

言語 : C#

## 15 囲碁カバディ部

香 川  
(高松)

村田 優斗 (5 年) 中野 将生 (4 年)  
桑村 真生 (3 年) 柿元 健 (教員)

### 1. システム概要

Web カメラを使って QR コードを読み取る。その後、用意したアルゴリズムでエージェントの行動を決定し、司令塔から伝達する。伝達する際、トランプは塩ビ管、三脚、両端クリップを組み合わせで作ったスタンドを使って提示する。また、エージェントの行動は GUI 操作で入力する。

### 2 GUI

フィールドを表示し、マウス操作で各エージェントの行動を入力する。

### 3. 機材

- PC
- Web カメラ
- 塩ビ管
- 両端クリップ
- 三脚

### 4. トランプの提示方法

トランプはスタンドに固定して 2 枚提示する。例を図に示す。



図 トランプの提示方法

### 5. 開発環境

OS: ArchLinux, Antergos

Language: Rust, TypeScript, Reason

Library/Framework: React.js, Electron

Editor: Neovim

ITS: GitHub

## 16 KCB

木 更 津

小島 五大 (3 年) 青山 大地 (3 年)  
岩井祐一郎 (3 年) 和崎 浩幸 (教員)

### 1. プログラム概要

競技中の視覚的なフィールド認識のための競技進行記録プログラム「きろぼん」で以下の画像のように記録を行います。そしてポイント取得については特定の情報入力に伴い現状状況での取得可能ポイントの中から最高ポイントを示してくれるポイント探索型プログラム「ぼねぼん」を用います。尚、ぼねぼんの技術を併用して相手チームのポイント取得の妨害箇所特定プログラム「じょきよぼん」も用いるつもりです。

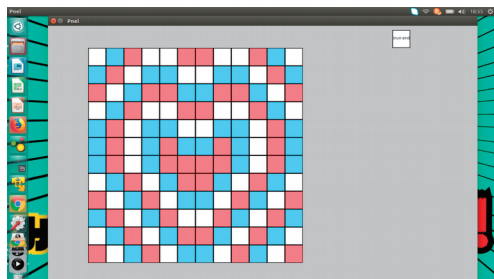


図 1. きろぼんでの競技進行記録の表示例

### 2. エージェントへの指示方法

トランプを用いて八方向の指示を行います。この際に行う指示は絵柄と演算方式を持ち合わせることによって 100 以上の表現方法を実現しました。これによって指示情報の漏洩を防ぐことに成功しました。

### 3. 予定競技進行

基本はぼねぼんが示すパネルを取得していき、じょきよぼんが高い数値を示した場合は除去を優先するというスタイルです。

### 4. 開発状況

開発言語: C 言語 Python Java

開発環境: Ubuntu Windows Processing3

## 17 巡る 8Byte

茨 城

小坂部恭輔（4年） 薄井 雅貴（4年）  
井上 将貴（3年） 安細 勉（教員）

### 1. システムの概要

QRコードを読み取るプログラムとエージェントの行動を決定するプログラムをそれぞれ用いる。

まず、QRコードをスキャナーで読み込み、OpenCVとZBarを使用したプログラムによって盤面のデータを取り込む。

また、エージェントの行動決定に当たっては、後述のアルゴリズムによって盤面の状況から最適と思われる行動を算出し、図1のようなGUIを通して司令塔に提示する。

### 2. 情報伝達方法

赤スートは移動、黒スートはタイル除去を表し、ジョーカーは停留を表す。そして、数字の値によって方向を表し、司令塔の情報をエージェントに伝える。

### 3. 行動決定アルゴリズム

最初に、相手より自分からの距離が近いマスの中で、高得点を一定ターン以内に取り又は奪うことが可能なパ

ターンを複数算出する。その後に、敵と味方の行動によるリスクを考慮して評価関数で評価し、行動を決定する。なお、評価関数の優先度を変更したアルゴリズムを複数のPCを用いて計算することで、最善手を求める。

### 4. 開発環境

開発言語：C++

使用ライブラリ：OpenCV, ZBar, DX ライブラリ

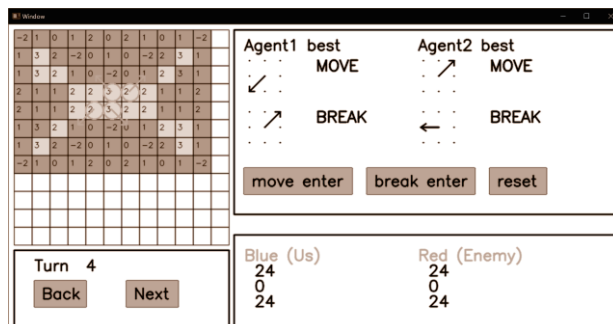


図1 GUIの例

## 18 ★このチーム名を決めるまでに90分かかりました★

豊 田

神取 大貴（5年） 鶴田 拓磨（4年）  
長坂 光将（3年） 藤原 賢二（教員）

### 1. システムの概要

競技内容は陣取りであるので、できるだけ効率よく得点を取得できるように、そのターンでの盤面の状況からエージェントの動きを決定する。また、敵チームのエージェントが味方チームの行動を妨害することや味方同士の移動先が重複することを防ぐために、エージェントの動きを決めるアルゴリズムを変えている。

### 2. エージェントのアルゴリズム

片方のエージェントは数手先の最善手を求めて、行動をする。加えて、相方の邪魔となる近い距離には、近づかないように行動する。他方のエージェントは、盤面の状況から、領域点を含め最大の得点を取得する方法を求める。

### 3. GUI

図のように、盤面状況とエージェントを表示する。緑の丸が各エージェントの位置で、色のついたパネルが各チー

ムの取得したパネルとなる。領域として取得したパネルは薄い色で表示される。味方は自動で動き、敵は緑の丸を動かすことで、エージェントの移動が行われる。

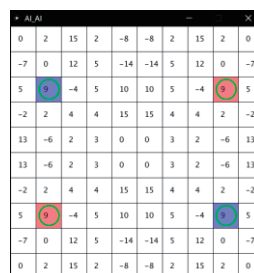


図 表示画面

### 4. 司令塔とエージェントの通信方法

通信には、司令塔側は主にトランプを使い、エージェント側はハンドサインを使用して、お互いの意思を伝える。

### 5. 開発環境

C++ , Processing(GUI用) , Visual Studio(AI用)

## 19 八代の三銃士

熊 本  
(八代)

古川 照英 (2年) 鶴本 尚己 (3年)  
豊世 光希 (3年) 小島 俊輔 (教員)

### 1. はじめに

今回の問題では、得点の大きいタイルを効率よく抑え、領域ポイントを得ていくことが重要になる。そこで、貪欲アルゴリズムを用いて解く方法と、盤面のデータをもとに解く方法の2つのプログラムを用いる。

### 2. 貪欲アルゴリズムを用いる方法

まず、盤面を3~5手先まで全探索し、敵にタイルを取られないか、得点の大きなタイルはどこか、領域ポイントを効率よく得るにはどの方向に進めばよいかなどを評価する評価関数を用いて判定していく。途中経過の得点を確認し、得点が低いものは早い段階で枝刈りし、高いものについては優先的に探索することで探索時間を節約する。また、視覚的にわかりやすいように盤面と解を一緒に出力する。

### 3. 盤面データをもとに解く方法

まず、様々なパターンの盤面データを、乱数を用いて自動で作成する。その盤面データをもとに、似た盤面があった場合、どの方向に進んだときに、得点が多く得られるかなどを判定する。似た複数の盤面を比較し、勝率の高かった方向へ進ませるようにすることで、盤面データによる偏りを防ぎ、より良い解を導く。

### 4. エージェントへの指示伝達方法

最初に方向についての数字をあたえ、次にトランプを用いて絵柄と数字の組み合わせにより行動する方向、停留を指示する。またトランプの絵柄の構成やハンドサインによりタイルを置くか相手のタイルを消去するかを指示する。

### 5. 開発環境

OS : Windows10 言語 : C/C++, python

ソフト : Visual Studio2017, Anaconda3

## 20 7 Months to Dice

仙 台  
(広瀬)

穀田 一真 (4年) 高橋 唯 (4年)  
早坂 太吾 (4年) 園田 潤 (教員)

### 1. システム

#### 1.1 全体構成

システムは、QRコードの読み取りを行うプログラム、フィールドを管理するプログラム、相手の行動を入力するプログラム、こちらの指示を決定するプログラムからなる。

#### 1.2 QRコードの読み取り

競技開始時に一度だけ使用する。QRコードを撮影して読み取りを行い、フィールド管理プログラムへ送信する。

#### 1.3 フィールドの管理

TCPサーバーとして、有線LANを用いて他プログラムと通信する。競技フィールドをゲームとして保管し、ほかのプログラムと連絡しあって結合する。また、行動の誤入力などを修正する機能を持つ。

#### 1.4 相手の行動の入力

相手の行動を入力し、フィールド管理プログラムに送信

する。実際の競技フィールドをwebカメラで撮影してPC上に表示し、フィールドをマス目ごとに区切ったものをオーバーレイ表示して、相手の動きを直感的に入力しやすくする。

#### 1.5 指示決定

フィールド管理プログラムと通信を行い、こちらのチームの指示を決定する。

### 2. 指示決定アルゴリズム

行動決定には、Alpha Zeroを用いる。Alpha Zeroとは、Googleの開発したゲームAIアルゴリズムであり、ゲームのルール以外の入力が必要とせず、自己対戦によって強化される。また、この他にも自分たちで考えたアルゴリズムを利用し、最も効果的なものを使用する。

### 3. 開発環境・使用ライブラリ

Visual Studio 2017、C++、OpenSiv3D、Boost、Python

## 21 高橋 n 段

### 仙 台 (名取)

高橋 諒大 (3 年) 鈴木 暢真 (4 年)  
菊地 輝 (3 年) 北島 宏之 (教員)

#### 1. はじめに

今回の競技は対人戦のため、機械学習を用いた評価及び探索を行った。また、その評価をもとに司令塔が指示を出し、相手の指示を入力しやすいような UI の製作を行った。

#### 2. アルゴリズム

##### 2.1 評価値について

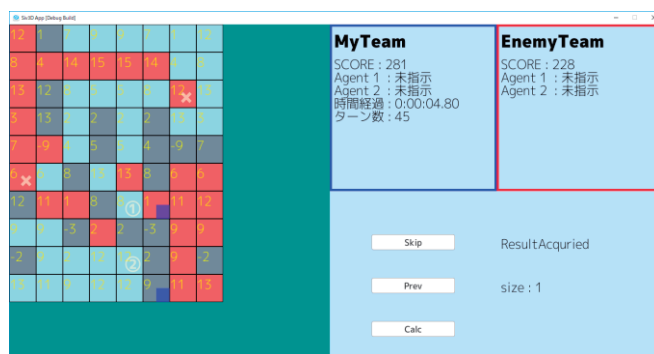
対戦データを利用した機械学習により大まかな評価値を作成するとともに、人の手を用いて調整を行うことで評価値を洗練した。

##### 2.2 探索部について

モンテカルロ木探索を利用してランダムに回数を重ねて探索し、効率的に点差を広げていく指示や、不利な状態から逆転する指示を列挙させた。

#### 3. UI について

UI は入力 of 正確性と速度を重視してミスが起きづらいように整えた。また、指示ミスを減らすために具体的なハンドサインが表示されるような UI を開発した。



#### 4. 開発環境

C++, VisualStudio2017, OpenSiv3D, Caffe, Tinydnn

## 22 すなぎも

### 群 馬

清水 怜央 (3 年) 大久保拓哉 (3 年)  
砂賀 開晴 (3 年) 木村 真也 (教員)

#### 1. はじめに

機械学習と純粋なアルゴリズムの 2 通りの方法を用意した。

#### 2. 機械学習

##### 2.1 手法 1

盤面情報を畳み込みニューラルネットワークの入力として与え、その盤面における評価値を出力する。

##### 2.2 手法 2

深層強化学習のアルゴリズム、Distributed Prioritized Experience Replay の中で紹介されていた Ape-X モデルを用いて優先順位付き経験再生による分散学習を行い、盤面の評価および、Dueling-DoubleDQN の出力する Q 値により、その盤面での行動を選択する。

#### 3. アルゴリズム

アルゴリズムは複数の種類から状況に応じて適宜変更する。

##### 3.1 手法 1

自エージェントの周囲をある程度の範囲で全探索を行い、最適な手を求める。相手のエージェントが領域を作りだしそうな場合はその領域によって加算される点数を考慮する。場合によっては全探索の結果を活かしつつ相手の領域を排除する。

##### 3.2 手法 2

相手エージェントの付近に自エージェントを配置することで相手の行動範囲を狭める。また、相手が移動する場所を予測してエージェントを可能な限り同じ位置に移動させることで相手の妨害を行う。

## 1. 概要

最適な行動の探索に時間を割き、エージェントへの意思疎通を短時間かつ正確に行うために、司令塔・エージェント間の意思疎通、前のターンの相手の行動の入力を迅速に行えるようにシステムを構成した。

## 2. 探索アルゴリズム

次のターンにおける味方チームの行動を探索するアルゴリズムを同時に複数動作させ、それぞれのアルゴリズムの解、もしくは最適と思われる解を選択して出力する。探索の際には、適宜相手のチームの行動も予測する。

## 3. GUI

味方同士の区別をしないことでエージェントを指

定させる行為を省き、司令塔が GUI に対して行う動作を選択と更新のみにした。また、指定を間違えた時のためにログをとって過去の盤面に戻れるようにした。GUI の表示例を図 1 に示す。

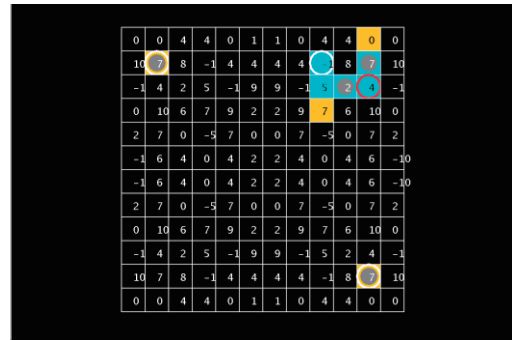


図 1 GUI

## 4. 開発環境

言語：Java、C++

エディタ：Vim、Visual Studio Code

## 1. システム概要

次の順序で処理を行う。③の後は①に戻る。

- ①プログラムがそのターンに最適な行動を求め(2.2)、画面に出力
- ②司令塔が指示を出し、エージェントが実際に行動
- ③両チームのエージェントの行動をプログラムに入力

## 2. システム構成

## 2.1 QR コードを読み込む部分

PC に付属のカメラを用いて、配布された QR コードを読み込み、フィールドの情報を取得する。

## 2.2 最適な行動を決める部分

各マスの重要度を遺伝子として扱い、遺伝的アルゴリズムを用いて、最適なマスの重みを決定する。この重みの値をもっとも大きく取れるように、次のターンの行動を決定する。

## 2.3 行動結果を入出力する GUI の部分

フィールドの情報をマス目上に表示し、どちらのチームが囲んでいる又はタイルを置いているマスなのかを色分けして表現する。図 1 は、表示の例である。実際の行動を反映するには、エージェントのいるマスをクリックし、次に行き先のマスをクリックすることで、移動を入力する。



## 3. 開発環境

言語：Java

統合開発環境：Eclipse

## 25 TMCIT アラカワ

都立  
(荒川)高澤 楽 (4年) 伊藤 裕也 (4年)  
児玉 暁憲 (4年) 齋藤 敏治 (教員)

## 1. はじめに

今年の競技部門は、マス目に区切られたフィールド上で以下に多くの陣地を占有することができるかを競い合う陣取りゲームを行う。

これに対して、私たちは、ミニマックスを用いたプログラムを作成した。

## 2. 内容

- ①初めに、指示された自チームの初期位置より、近い位置のフィールド内の外周に向かいながら、マスを占有していく。
- ②外周についたならば、その後、フィールドの外周に沿って、フィールド全体を囲むように進んでいく。
- ③フィールド全体の外周を囲み終わったなら、囲み終えた場所から、ミニマックス法のアルゴリズムのもと、発生する損失が最小となるようにフィールド内

の最適なルートを計算し、そのルートを通る。



## 3. 開発環境

- Java

## 26 NON STOP

## 弓削商船

金谷 咲弥 (1年) 小井川秀斗 (1年)  
行友 大貴 (1年) 長尾 和彦 (教員)

## 1. はじめに

今回の競技部門は、タイルポイントの高いパネルを優先的に自チームのタイルにし、タイルを大きく囲んで領域全体を取得する陣取りゲームである。ゲーム進行は、司令塔がフィールドの状態（配置状態、エージェントの位置）を元に、次の動作を算出し、同時に提出しなければならない。また司令塔からエージェントへの指示はトランプを用いるため、第三者の盗聴が可能となる。

## 2. システム開発の要件定義

競技の解析から、以下の仕様を満たす戦略が必要である。

- 司令塔の入力システム：エージェントからの状態変化の伝達、カメラ入力は困難である。司令塔が単独で入力確認することができる操作性の高い GUI の開発を行なった。
- 符号化の最適化：トランプを用いた指示は手間がかかるため、できるだけ簡潔に進めたほうが望ましい。そのため、

数パターンの方式の符号化を準備した。他チームの暗号解析については、競技中の解析は困難であるため、除外する。

- QRコードによるデータ取得：カメラ入力に対応。
- 戦略アルゴリズム：シミュレータで得た知見を反映した。

## 3. 開発環境

MacbookAir (Python3, macOS10.12.4) + iPad を予定

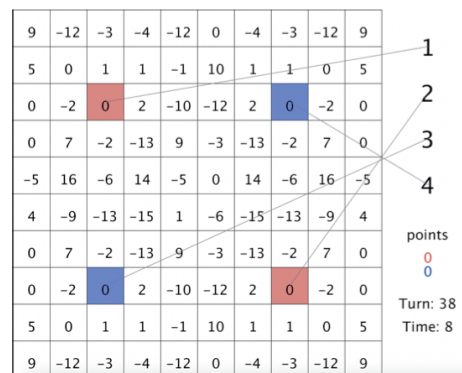


図1 シミュレータ

### 1. はじめに

今回の問題では、マスへのデータ入力、進むマスを決めるアルゴリズム、データ出力の 3 つの処理に分けて考えることができる。

### 2. 入力

QR コードを web カメラで撮影し、OpenCV で画像の抽出を行い、ZBar を用いて QR コードを文字列に変換して入力を完了する。

### 3. アルゴリズム

最初にマスのポイントの大きさ、味方のエージェントからマスまでの距離、マスの占有状況、敵の領域を作っているマスかどうかから、評価値を計算する。次に評価値から目標のマスを決める。4 人のエージェントから目標のマスへのマイナスのマスを通らない最短ルートをそれぞれ動的計画法により求める。最短手順が相手の方が少ない場合、

目標マスを決め直し、味方の方が少ない場合、味方エージェントの余裕手数を考慮して、より多くのポイントを獲得できる経路を求めて進むマスを決定する。

### 4. 出力

自分・相手の手の入力や、エージェントへの指示、状況確認を円滑に行うためには、分かり易い GUI である必要がある。今回の GUI は、エージェントの位置、マスの得点、目標マスと経路、総合得点、ターン数、エージェントへの指示方法が分かるように構成している。

### 5. 開発環境

【OS】Windows7

【IDE】VisualStudio2017

【Language】C++, C#

【Library】ZBar, OpenCV

【Other】Unity

### 1. はじめに

今回の競技では敵チームより多くの点数を獲得するために、限られた時間で点数配置や進行状況を考慮した最適な行動を速く算出することが重要となる。

### 2. システム概要

#### 2.1 データの入力

各マスの点数およびエージェントの配置方法を配布される QR コードから取得する。QR コードの読み取りにスキャナを用いることにより、誤りのない情報の取得を目指す。取得した情報はテキストファイルとして保存する。

#### 2.2 司令塔の指示決定アルゴリズム

行動可能な 2 手先までの行動パターンの中から、ミニマックス法を基にしたアルゴリズムを用いてエージェントの最適な行動を算出する。自チームが有利となる行動パターンが 2 手先までの行動パターンの中に存在しない場合

は、存在する n 手先までの行動パターンを算出する。

#### 2.3 データの出力・更新

算出した最適な 2 手先までの行動をパソコンの画面に表示する。

相手チームと自チームの意思表示が終わり次第直ちに両チームの 2 行動の入力を行う。その入力に応じて、新たに最適な行動を算出する。

### 3. エージェントへの指示伝達

大会規定の A4 サイズのトランプおよびハンドサインを用いてエージェントへの指示を行う。

### 4. 開発環境

OS : Windows 10

Microsoft Visual Studio Express 2015 for Windows  
Desktop

開発言語 : C++

29

1/2 の確率で右上に進む  
チーム

都 城

萩原 拓望 (2年) 瀬尾 尚寛 (3年)  
片山 北翔 (2年) 丸田 要 (教員)

## 1. はじめに

今回は、探索アルゴリズムを使い、距離と得点から短い距離で高い点数を取れるタイルを導くプログラムを開発する。

## 2. 全体の流れ

タイルポイントを  $3 \times 3$  のグループに切り取り、範囲内のポイントを足した評価値を算出。これを繰り返し、評価値の高い戦略領域とし、そこからエージェントまでの最短距離を求める。評価値をエージェントからの距離で割り、再度評価値を算出。これにより一番値が高かった戦略領域を囲うようにエージェントを動かす。

## 3. 得点の計算

味方が踏んだタイルの座標を二次元配列に格納しておき、戦略領域得点関数で得点を算出。味方が踏んだタイルの座標の値を 1、敵が踏んだタイルの座標の値を -1、そ

れ以外の誰も占領していないタイルの座標のところを 0 として記録していく。

誰も占領していないタイルの座標を 2 に置き換えていくと、領域ポイントは 0 のまま残るので絶対値にして計算する。

領域範囲は上記の方法で 1 という値で残るので、そのままの値で計算する。

## 4. 開発環境

- Microsoft Visual Studio 2017
- DxDLib



30 1/0

都 立  
(品川)和田 靖広 (4年) 高松 健 (4年)  
若山 裕輝 (2年) 福永 修一 (教員)

## 1. はじめに

今回の競技は、得点が割り振られたマスを取り合い、最終的に相手よりも高い得点を得ることを目的とした競技である。構成するシステムは、マスの得点とエージェントの初期位置をデータ化し、ターン毎にエージェントの行動を反映する入力部分、エージェントの次の行動を決める処理部分、決定した行動を表示する出力部分によって構成される。それぞれの処理について以下に説明する。

## 2. 入力：マスの得点とエージェントの位置の取得

入力部分は各マスの得点とエージェントの初期位置を入力する部分と、ターン毎のエージェントの行動を反映する部分に分けられる。各マスの得点と初期位置に関しては QR コードが配布されるので、Web カメラを用いてこれを読み込みデータ化する。ターン毎の行動は目視で確認し、手入力で反映させる。行動の誤認によって実際とは異なる状

態で処理が行われてしまう可能性があるため、一手前の状態に戻り、入力しなおすことができる GUI を用意する。

## 3. 処理：アルゴリズムについて

今回は、行動の評価が複雑であるためビームサーチを用いた探索は難しい。そこで、モンテカルロ木探索を用いて確率的に次の行動を決定する。また、並行して機械学習を行い、勝率の高い行動が得られた場合はこの行動を選択する。

## 4. 出力：行動と指示の紐づけについて

処理部分で決定した行動は、指示で用いるトランプの色と数字で出力される。これにより、速やかにエージェントに指示を行うことができる。

## 5. 開発環境

言語：C++, Java, Python3  
環境：Sublime Text 3, Eclipse

## 31 サレジオ情報局秘密情報部

### サレジオ

岡江 仁 (3年) 谷本 陽祐 (3年)  
和泉 怜恩 (2年) 宇都木修一 (教員)

#### 1. システムの概要

本システムは、探索部、入力部および出力・再入力部に分かれている。

#### 2. 各システムの説明

##### 2.1 入力部

入力部は、予め運営側から渡される QR コードを解読し、QR コードに格納されているフィールドの情報を探索部と出力・再入力部に転送する。

##### 2.2 探索部

探索部は、入力部から出力・再入力部から転送されたフィールドの情報をを使用して、フィールド上でエージェントが進むべき場所の提案を行う。エージェントが進むべき場所については、フィールドの初期状態をルート、その後予測されるフィールドの状態をノードとした木構造を考え、これに対しマスのポイント、領域ポイント、相手チームの動

きと相手チームの妨害を要素とした探索を行い、決定する。

##### 2.3 出力・再入力部

出力・再入力部は入力部から転送されたフィールドの情報を、司令塔が認識しやすくなるように表示する。また、探索部から転送された最良経路も司令塔が認識しやすくなるように表示し、司令塔がエージェントに対して行う指示を決定する材料とする。また、1 ターンのうちのエージェントの動きやタイルの状態の変化を入力できるようにし、このフィールドの更新情報を、再入力として探索部へ転送する。

#### 3. 使用言語・ライブラリ

入力部:Python, pyZbar, openCV

探索部:C++

出力部:Electron, Typescript, Reacto

## 32 高専におにはいません

### 石 川

河上 悠輝 (4年) 大野木 駿 (4年)  
北島 果歩 (4年) 小村良太郎 (教員)

#### 1. GUI について

Processing を用いて盤面の情報と自チーム、相手チームのエージェントの位置情報を表示することができる。また、毎ターンその時点での得点を計算させることができる。図 1 に作成した GUI を示す。

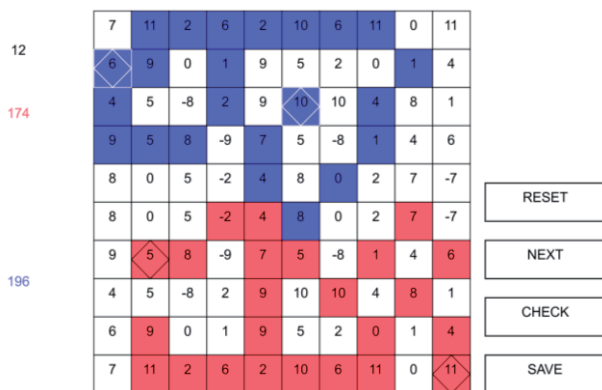


図 1 司令塔用に開発した GUI 画面の一例

#### 2. 司令塔とエージェントの通信手段について

##### 2.1 序盤の通信手段について

確実に指示を伝えるときに、相手に変換方式を解読されないようにするためにハンドジェスチャーを用いて指示を出す。

##### 2.2 中盤及び終盤の通信手段について

相手に指示を知られたくない状態では、トランプを用いて何試合しても相手に見破られないセキュアな方法で通信する。また、トランプを安定して表示するための道具を用いる。

#### 3. 主な作戦について

GUI 上で試行を繰り返した結果から序盤、中盤、終盤の 3 部分に分けそれぞれにおいて決定するアルゴリズムを考える。また、一部分においては手動により行う。

## 33 廻レ! 限界高専生

長岡

布施 智進 (3年) 山坂 昌司 (3年)  
矢野 敦大 (1年) 竹部 啓輔 (教員)

## 1. はじめに

今回の競技部門の内容は, 相手より得点の大きいマスを取り, 囲むというものである. これをプログラムによって, 最適解を求める.

## 2. QR コードの読み取り

Web カメラを用いて, ZBar, OpenCV によって生成したアプリで QR コードの読み取りを行う. 取得した文字列をソルバー部にパイプで渡し配列に格納する.

## 3. 処理・進行

主に,  $\alpha$ - $\beta$  法, ビームサーチなどのアルゴリズムを用い, 最適解を導出する. 序盤は静的評価, 定石から展開できるように思考ルーチンを構築している. 1 ターンの制限時間が最初は 30 秒, その後は 15 秒しかないため処理の時間を 5 秒に絞り, その後各エージェントへの伝達は一人当たりトランプ, ハンドサインを用いて 3 秒で伝達する.

## 4. GUI

Siv3D を用いて描画する. 直感的に操作することができるようにした.

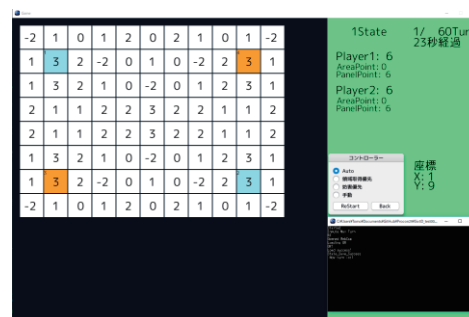


図 1 開発中の GUI

## 5. 開発環境

言語: C++, Python3

IDE: Visual Studio 2017, Anaconda,

ライブラリ: Siv3D, ZBar, OpenCV, Boost

## 34 俺たちがソルバー

熊本  
(熊本)末永 和大 (4年) 扇塚 和希 (3年)  
鶴田 惟月 (3年) 孫 寧平 (教員)

## 1. はじめに

今回の競技は対戦型となっており, 相手の手によって次にすべき行動が大きく異なってくる. よって, ターンごとに探索を行い, 次にすべき行動を決定する.

## 2. データの入力

Web カメラを用いて QR コードを読み込み, データをファイルとして保存する.

## 3. 探索アルゴリズム

各マスの点数を格納した配列とエージェントの座標を元にビームサーチによる探索を行う. 各マスを埋めるより, 囲んだ点数がマイナスであるマスもプラスの得点となるため, より効率的にマスを囲むことを基本方針する. ビームサーチにおける評価関数では相手エージェントの座標, 次に行う各行動でのスコア, 相手のスコアなど様々な項目に対して評価を行う.

## 4. GUI

自分の領域や各エージェント座標など, フィールドの状況が直感的に分かるような GUI を作成する. また, 次に移動するマスも同時に GUI に表示することで迅速にエージェントへ指示を遅れるようにする.

## 5. エージェントへの指示伝達

ハンドサイン等の意味が相手にバレにくいように一つの行動に対して複数のサインを組み合わせ, エージェントへ指示を送る.

## 6. 開発環境

OS: Windows 10

IDE・Editor: Visual Studio, Sublime Text

言語: C++, Python3

ライブラリ: Siv3D, Boost

## 1. アプローチ

(1)はじめに QR コードを読み取り、検出された文字列をパースし、盤面の大きさ、ポイントの配分、エージェントの初期位置から盤面を作成する。

(2)自分の周囲 8 方向の設定されたポイント数の中で最大のものを検出し、獲得ポイントの一番多いマス进行判断する。

(3)一番獲得ポイントの多いマスへ移動することをエージェントの最良の行動とし、AI が次の一手をサジェストする。

(4)司令塔がエージェントに指示を出し、動いた後の状況を AI に教える。

(5) (2)から(4)までを繰り返す

## 1. はじめに

今回の競技は囲碁と似たゲームであるが、違いとして同時手番ゲームであり先手や後手といった概念が存在しないとといったことなどがあげられる。敵チームの妨害も可能であるため、それを考慮したソルバーの設計を考える。

## 2. QR コードの読み取り

QR コードは Web カメラを用いて読み取り、OpenCV と ZBar を用いてデータ化し取り込む。

## 3. 指示決定アルゴリズム

盤面をノードとしたゲーム木についてビームサーチを行うことで、次の手を決定する。サーチ幅、サーチ長については開発しながら適宜チューニングする。自チーム・敵チームそれぞれの評価関数の差によって、盤面の評価が決定する。

## 4. 意思伝達手段

主に野球で使われるブロックサインを応用し、ボディランゲージとトランプを組み合わせで指示を行う。これは体の各部位に命令を割り当て、触った部位で指示を行うものだが、単純にその部位に触るだけでは通信の傍受が容易であるため工夫を施す。この指示を司令塔側の GUI アプリに表示することで、負担軽減とミスの予防を図る。また、エージェントと司令官で綿密な事前練習を行い、エージェント側の意思表示で発生しうるミスを減らす。

## 5. 開発環境

【OS】Windows10

【言語】C++, C#

【エディタ】Visual Studio 2017, Visual Studio Code

【ライブラリ】Eigen, OpenCV, Boost, Zbar

## 37

陣地行動  
— 人力版バトロワ —

有 明

富本 優介 (5年) 谷 廣 (4年)  
石川 順平 (5年) 森山 英明 (教員)

## 1. はじめに

今回、「巡りマス」をプレイするにあたり、python とそのライブラリである Keras を使い、AlphaGo Zero のアルゴリズムをベースにして「巡りマス」AI を実装する。実装した AI によりエージェントの行動を決定する。AlphaGo Zero のアルゴリズムは、人間の対戦データを学習に使用しないため「巡りマス」のような過去の対戦データがないゲーム AI の実装に向いていると考える。

## 2. アルゴリズム

AlphaGo Zero では、探索木アルゴリズムとしてモンテカルロ木探索アルゴリズムを使用している。しかし、今回の「巡りマス」は同時進行ゲームであるので、探索木のアルゴリズムを同時進行ゲーム用に変更する。変更するアルゴリズムの動作は、以下のようにする。

まず、探索木の生成の際に自分の合法手と相手の合法手

の組を枝とする。次に、生成された全てノードについて機械学習により勝率を計算する。その後、互いの合法手を戦略の組、UCB1 値を利得とし、ナッシュ均衡となる混合戦略を近似する。この混合戦略に基づいて探索するノードを確率的に計算し、計算によって選択されたノードの勝率を機械学習によって求める。これらの動作を一定回数繰り返した後、探索木のある地点の深さの各ノードについて互いの合法手を戦略、標本平均利得を利得とし、ナッシュ均衡となる混合戦略を近似、これに基づき着手を選択する。

## 3. 開発環境

言語:python ライブラリ:tensorflow, Keras

## 4. 参考文献

大町洋・池田心 (2012)「同時進行ゲームのためのモンテカルロ木探索」、ゲームプログラミングワークショップ 2012 論文集, 2012 (6), pp197-204.

## 38

## 自軍パネル破壊はテンポロス

福 井

永田 篤矢 (4年) 谷川 友基 (4年)  
河野 智弘 (4年) 村田 知也 (教員)

## 1. はじめに

今回の開発するシステムを構成する要素としては、主に「盤面情報の入力」、「アルゴリズムの処理」、「必要な情報の表示」の三段階である。それぞれの処理については以下に説明を記す。

## 2. 盤面情報の入力

まず、盤面の情報は QR コードによって渡されるので、カメラから情報を読み取りテキストファイルとしてデータを保存する。プログラムはこのテキストファイルを読み込み、盤面を構成する。プレイヤーの移動の入力はマウスの入力を用いており、司令官が手動で入力を行う。

## 3. アルゴリズムの処理

行う処理としては、自陣のエージェントがどこに動けば最も高い点数が獲得できるか周辺を探索するこ

とにより、より高い点数を獲得することを狙うというものである。

## 4. 必要な情報の出力

入力が行われた時点で盤面の情報と現在獲得している点数を出力し、計算が終わった時に次の一手をどこにすべきかを表示する。

## 5. 開発環境

言語: C++, C#, JavaScript

環境: Visual Studio 2012、

Visual Studio Community 2017

ライブラリ:

「<https://www.shuwasystem.co.jp/support/7980html/2118.html>」にある GameLib2012、  
.Net framework 標準ライブラリ、Node.js

## 39 太陽先輩

## 北九州

井上 航 (3年) 森 友希 (3年)  
窪田 秀哉 (3年) 松久保 潤 (教員)

### 1. はじめに

今年の競技部門では、マス目に区切られたフィールド上で、いかに多くの陣地を占有できるかを競う陣取りゲームを実施します。

構成するシステムは、  
入力を QR コードで読み込む部分、  
人がどう動くかを考える部分、  
考えたのを伝える部分です。

### 2. データの取得

QR コードで競技フィールドの縦と横のマス目、  
第一行目の左から順にそれぞれのマス目の点数、  
二つの駒の初期位置のデータが与えられます。  
その QR コードを読み込んでそれらの情報を取得します。

### 3. 処理: アルゴリズムについて

ビームサーチを行います。

- ・相手の駒の動きを読んでいては探索時間が足りないため、相手の駒ではなく自分の駒だけ 5 手動かす方法のうち、評価値が最大のものを選びます。
- ・自分の 2 つの駒を独立に考えます。
- ・自分の陣地で囲ってもすぐに壊されてしまうため、最初のほうはあまり囲うことを考えません。

### 4. 伝達方法

ランプの数字を用いてテンキーの配置を利用し、方向を出力します。

司令官が二人のエージェントに方向を指示するとき、ランプのスタートの色で二人のどちらかを区別します。

### 5. 開発環境

【OS】 windows 7, 8, 10 【コンパイラ】 visual c++  
【エディタ】 サクラエディタ 【言語】 C++  
【ライブラリ】 Dxxlib

## 40 HelloWorld

## 国 際

鈴木 大志 (4年) 中西 勇人 (3年)  
東本 一希 (3年) 田村 景明 (教員)

### 1. システム概要

今回の競技部門の問題に対して、あらかじめシミュレーションを行い機械に学習させ、試合時に学習データ内から最適解を検索するアルゴリズムを考案。また、試合の状況や解の内容の入出力を行う GUI を作成した。

### 2. システム詳細

#### 2.1 学習

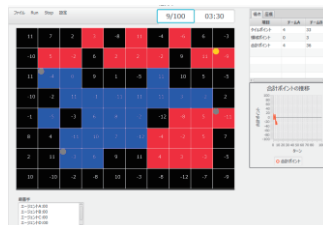
シミュレーションソフトを用いて学習データの作成を行う。作成したデータは扱いやすいように変換された後、データベースへと整理して記録される。

#### 2.2 解の出力

試合の状態から検索条件を作成。その後、データベース内から条件が一致するデータの検索を行う。この時に一致

するデータがなければ条件に限りなく一致するものを検索。最後に検索結果をディスプレイに出力する。

### 3. GUI



状況の入力や解の出力に使用される GUI

### 4. 開発環境

【言語】 C++ , Python3, JAVA  
【IDE】 Visual Studio 2017, atom, pycharm  
【ライブラリ】 Boost, SQLite, JavaFX

## 41

司令塔の指示を優先する？  
さてはオメー人力だな

## 宇 部

秦 彩歌（4年）堀江 裕貴（4年）  
朝顔 栞（1年）江原 史朗（教員）

## 1. はじめに

今回の競技は、2つのチームが1つのフィールドに存在するマスに対して同時に操作を行い、より多くの点を得るものである。配布されるフィールドの得点情報より、より多くのポイントを取得することを目指す。

## 2. 行動決定アルゴリズム

以下に、手番中に行う行動の決定に用いるアルゴリズムの概要を示す。

- (1) 配布された QR コードより、フィールド上のマスの得点情報をプログラム上に展開する。
- (2) 得点の分布と競技の性質、また各チームが取得済みのマスの得点・位置情報を元に、各マスの重み（重要度）を計算する。
- (3) (2)によって得られた重み情報を元に、取りうる手の

中から最善と思われる手を決定する。

- (4) 手番の経過によって各チームの暫定取得マスや点数が変化するため、それらの情報を更新し(2), (3)を繰り返す。

以上の要領で現在取りうる最善手を導き出す。

尚、今回の競技では与えられる時間が非常に短いため、司令塔となった選手が素早く画面操作をできるように表示される GUI のデザインについても重要視する。

直感的に操作可能な GUI 上に、前述したアルゴリズムによって導き出された結果を次々と盤面上に表示する。

## 3. 開発環境

Windows10, ArchLinux

## 42

## サウンド・オブ・ハイドロゲン 鹿児島

## 鹿 児 島

東 誠太（4年）市川 知春（4年）  
引地 涼（1年）武田 和大（教員）

## 1. はじめに

今回、鹿児島工業高等専門学校は、全国高専プログラミングコンテスト競技部門に向けて、移動方向をアルゴリズムで導出し、適切な指示を司令塔がエージェントに対し送るサポートをするシステムを開発している。

## 2. システム全体の流れ

## 2.1 QR コードの読み込みについて

QR コード用のライブラリを使用し、QR コードを.pqr(高専プロコン用 QR コードファイル)に出力する。

## 2.2 アルゴリズムと解法表示プログラムの連携について

GUI アプリケーションのプラグインとしてアルゴリズムを組み込んだボットによって、移動方向を決定する。

アルゴリズムを変更するときは、導入するボットを差し替えるだけで効率的にアルゴリズムをテストすることが可能となる。

## 2.3 アルゴリズムについて

複数のアルゴリズムを人力及び機械学習により、作成、対戦させ、より良いアルゴリズムを使用する。

人力で作成するプログラムは斜め移動を中心としたものとするので、エリアポイントを取得できるようにする。

## 2.4 解答の出力について

アルゴリズムを用いて導き出した解法を見やすい GUI アプリケーションとして表示する。また、司令塔がエージェントに対して送る信号を画面上に表示し司令塔のミスを防ぐようにしている。

## 3 開発環境

【OS】 Windows10, macOS High Sierra

【IDE】 Microsoft Visual Studio 2017

【Language】 C, C#, Python

【Editor】 Visual Studio Code, Vim

## 43 考えたくないでござる

神戸市立

宇貞 勇輝（4年）黒井 大輔（4年）  
上村 裕（3年）朝倉 義裕（教員）

### 1. システム概要

QRコードを読み取り、得た情報からGUIを作成。次に、敵のエージェント2人の位置を随時入力し、その状況に最適なエージェント2人の行動を出力させ、司令塔がその手をエージェント2人にトランプを用い伝達する。

### 2. 行動の決定方法

現在の局面から探索を行い、次の局面を評価した値により行動を決定する。

局面の評価は、あらかじめ学習されたニューラルネットワークを用いる。このニューラルネットワークに必要な情報を入力し、次のすべての1手についての評価値を出力させる。そして違反手をフィルタリングした後、それぞれの局面からさらに次のすべての手を評価する。一定の深さまで探索した後、最善手を決定する。

### 3. ニューラルネットワークの概要

入力パラメータはマスの大きさ・マスの点数・敵と味方のエージェントの位置・敵と味方の於いてあるパネルの位置等からなる。盤面の状態を画像としてネットワークに入力し、適切なフィルタで畳み込み、最終的に値を出力することによって、次のすべての手を評価する仕組みとなっている。

### 4. ニューラルネットワークの学習方法

ニューラルネットワークは自己対戦やランダムな局面に対する探索の結果によって強化学習を行う。具体的にはある局面から深く探索し、その結果を浅い探索でも出せるように学習を行う。

### 5. 開発環境

言語：Python3

IDE：Visual Studio

仕様ライブラリ：NumPy, wxPython, Chainer

## 44 囲碁作戦

徳 山

杉本 航（3年）中谷 響（3年）  
伊藤 大志（3年）力 規晃（教員）

### 1. 今年度競技部門の概要

今年の競技内容で特徴的な点はターン制で1ターンが基本的に15秒であること、そして毎ゲームでゲーム盤が変わることである。

そこで私たちはプログラムの「速さ」と「適応力」に注目した。この二つを考慮したプログラムでなければたとえ正しい答えを導き出せたとしても勝利するには程遠いと考えた。

### 2. 作ったプログラムのコンセプト

1章で述べた2つの要素を考慮したプログラムをモンテカルロ木探索という探索アルゴリズムを用いて作る。

15秒未満で導き出せる「速さ」があり、かつプログラムにはどんな盤面でも適応できる「適応力」を兼ね備えたのが私たちのプログラムである。

### 3. 行動指示

今大会において求められることはプログラミング能力だけではない。プログラムが導き出した答えを伝える行動指示の方法にも一工夫が必要である。できるだけ迅速に行動指示を伝えられなければならない。

具体的に言えば、よく使う命令はできるだけ簡潔になるよう、指示に冗長さを無くし、伝える情報を少なくすることにより、情報伝達を迅速に行うようにする。

### 4. 開発環境

言語：C++, C

OS：Windows10

ライブラリ：OpenCV

## 45 岐阜高専スイーツ同好会

岐 阜

利根 悠司 (4 年) 柴野 和樹 (4 年)  
川瀬 敬弘 (4 年) 廣瀬 康之 (教員)

## 1. はじめに

2 チームによる対戦形式の問題である。1 ターンの戦略決定にかけられる時間が短いため、基本的には予め用意したプログラムの指示に従う。

また、戦略伝達の際は相手チームに看破されないように最低限の暗号化を施す。そのエンコードをはじめとする、人間が絡むパートでの本番のミスを防ぐために十分な訓練を重ねる。

## 2. アルゴリズム

盤面の各マスには得点が振られており、静的な評価関数がある程度組みやすくなっている。図 1 のように用意したビジュアライザを適宜使いながら、評価関数の内容を自分と相手の双方のチームについて検討し、それに基づいたヒューリスティック探索を行う。

## 3. 開発環境

言語: Java8

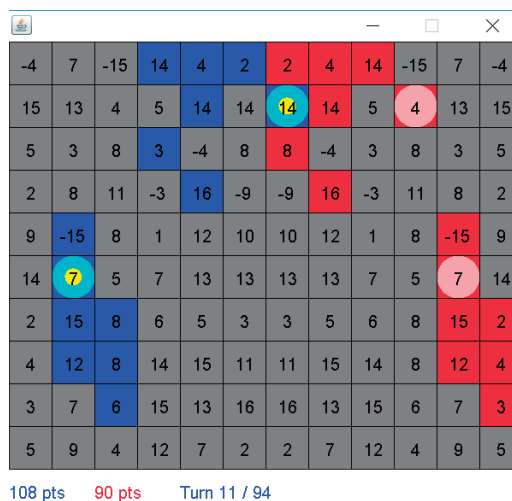


図 1 ビジュアライザ

46 山吹色のお菓子  
～その時パネルが動いた～

佐 世 保

小西 宏樹 (4 年) 田中 樹 (4 年)  
今井 壮志 (4 年) 嶋田 英樹 (教員)

## 1. 概要

両チームの司令塔がエージェントに最終ターンまでランダムに指示を出し、何度もそれを繰り返すことによって盤面の評価をするモンテカルロ木探索を軸とした指示決定アルゴリズムを実装する。

## 2. モンテカルロ木探索

## 2.1 基本動作

基本動作は、次の動作 1. ～ 4. の繰り返しである。

1. 可能な手から一手選び、木を下りていく
2. 末端で新たなノードを作成
3. シミュレーションを一回行う (プレイアウト)
4. 結果を更新する

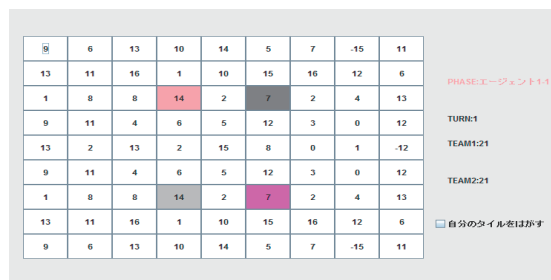
## 2.2 手の選択

探索にかけることのできる時間は 10 ～ 15 秒と非常に短く、完全にランダムなモンテカルロ木探索では満足で

きる探索を行うことができない。そこで、UCB・UCT という考えを導入する。これは一手の価値を、プレイアウトでの勝率と期待値で決定するもので、それをもとに探索木を下りて、探索にかかる時間を短くする。

## 3. GUI

コンピュータの操作にかかる時間を短縮するため、直観的な操作が可能になる GUI を実装する。



## 4. 開発環境

言語: C/C++ Java IDE: Visual Studio 2015 Eclipse

## 47 極めて普通のプログラマ

小山

高井 淳光 (4年) 三村 泰世 (3年)  
安田 光希 (2年) 平田 克己 (教員)

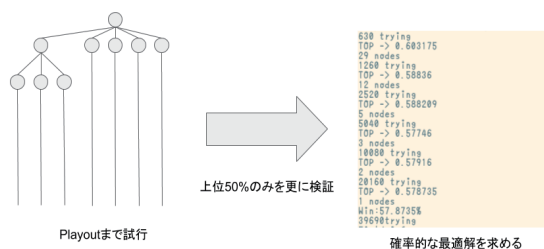
### 1. はじめに

今回の競技では、それぞれの陣営の得点計算は容易ではあるが、それを直接盤面の評価に用いるとエージェントが短絡的な行動をしやすくなる。これを防ぐためにはエージェント同士の位置関係、周囲の非支配パネルの点数が非常に重要になってくる。そこで本システムでは主の探索アルゴリズムとしてモンテカルロ木探索を用いることにした。

### 2. 戦略

今回の競技では一手のパターンが膨大で、単純な木探索アルゴリズムでは探索量が指数関数的に増加する。また、評価関数は複雑になり精度向上も難しくなる。一方、本システムで採用したモンテカルロ木探索は、シミュレーションにより確率的な最適解を選択する手法であり、評価関数を必要としないため比較的、精度・計算時間ともに高い水準に到達することが期待される。また、斜め移動で囲いを

作ることによって効率的に得点を得られる点に着目し、現在位置から最小の囲いを作るような動作を列挙して得られた得点を評価指標とする手法も実装した。本システムでは高精度な最適解予測のために、モンテカルロ木探索と上記の独自アルゴリズムとを序盤や中盤などの時間別、あるいはエージェント間の距離によって使い分けるようにしている。



図：モンテカルロ木探索の実行結果

### 3. 開発環境

OS: Linux; IDE: Emacs, Geany; 言語: C++, C#, Java;

## 48 指示出す阿呆に動く阿呆、同じ阿呆なら……

奈良

三野 天羽 (4年) 藤本 祥 (4年)  
比嘉 隆貴 (2年) 岩田 大志 (教員)

### 1. フィールド状態の入力

フィールド状態の入力は、以下の図に示すように GUI を用いて直感的に行えるようにする。

|    |    |    |    |    |
|----|----|----|----|----|
| 15 | -7 | 6  | -7 | 15 |
| 10 | 6  | -8 | 6  | 10 |
| 8  | 7  | 12 | 7  | 8  |
| 10 | 6  | -8 | 6  | 10 |
| 15 | -7 | 6  | -7 | 15 |

ただし、試合開始時のフィールド状態は配布される QR コードをウェブカメラで読み取り、取得するものとする。

### 2. アルゴリズム

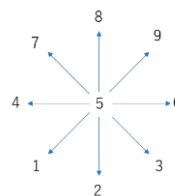
入力されたフィールドの状態から、エージェントの取りうる行動をビームサーチで探索する。

その際、2人いるエージェントの動きを別々に考え探索を行うことで、計算量を少なくする。

また、負の数がポイントして設定されているマスなどの探索を枝切りすることで高速化を目指す。

### 3. エージェントへの指示伝達方法

エージェントの動く方向は図に示すように、数字で表す。



司令塔は事前にエージェントと共有した鍵(数字)と、エージェントが直近に動いた方向の数字、今回指示する方向の数字等をもとに行動する。

### 4. 開発環境

開発環境 : IntelliJ IDEA, VS Code

言語 : Kotlin, C++, Python

## 49 我ら身代わり隊

富 山  
(射水)

岩田 有喜 (2年) 高畠 敦 (2年)  
大道 太陽 (1年) 山口 晃史 (教員)

### 1. システム概要

本プログラムではあらかじめ機械学習によって作成したモデルを使用して、『巡りマス』におけるエージェントへの指示を決定する。

### 2. プログラム

#### 2.1 機械学習のモデルの作成

モデルを作成するためのデータを収集は、モンテカルロ木探索を用いてエージェントへの指示を決定し、コンピュータ自身と対戦させることで行う。これにより機械が自動でデータを作ることができ、データ収集を効率よく行えるようにした。また探索に評価値を数手先の点数に設定し、プログラムがあたかも未来予測を行っているような手を選ぶようにしている。

#### 2.2 『巡りマス』対戦プログラム

QR コードから読み取った点数の配置やエージェントの

位置、およびタイルの位置をもとにプログラムがエージェントへの指示を決定する。エージェントやタイルは絶えず移動するので、GUI を用いて追跡操作を容易に行えるよう工夫した。

### 3. 開発環境

言語: C++, Python

IDE: Visual studio 2017 Community, Spyder

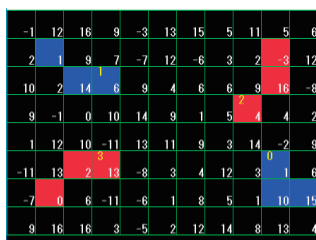


図1: コンピュータ自身の対戦の様子

赤: 敵のタイル 青: 味方のタイル

白文字: マスの点数 黄文字: エージェントの番号

## 50 鶴岡マス取部

鶴 岡

村山 肇 (4年) 匹田 祐平 (4年)  
五十嵐拓真 (2年) 金 帝演 (教員)

### 1. 概要

本アプリケーションは指示決定アルゴリズムとエージェントへの指示伝達に構成されている。

司令塔がエージェントへ指示を決定するアルゴリズムは3つのモード(領域確保、攻撃、防御)を使い分けることで、相手の戦略に対して柔軟に対処することができる。司令塔からエージェントへの指示伝達は、エージェントが瞬時に理解できるように工夫をしている。

### 2. 指示決定アルゴリズム

司令塔がエージェントへの指示を決定するアルゴリズムは3つのモードで構成されている。領域確保モードは得点の高い領域を判断し、自領域として確保するモードである。攻撃モードは相手チームに領域を確保させないモードである。相手領域のタイルを除去することで相手に領域点数が取れないようにする。防御モードは自領域を守るモー

ドである。相手エージェントが自領域を破壊する前に相手エージェントと自領域の間にタイルを置いて自領域を守るモードである。これらのモードは相手の動きから判断する。

### 3. エージェントへの指示伝達方法

上記のアルゴリズムから決定された指示をエージェントへ伝達する方法について述べる。

エージェントへ指示内容は相手が分からないようにし、我々のエージェントは瞬時に理解できるように暗号化する。具体的にはターン数と方位を2進数に変換したものをXORする。また、2進数(0と1)表現はカードの裏と表、または偶数と奇数で表現する。

### 4. 開発環境

開発環境 : Visual Studio

OS : Windows 10

言語 : C#

## 51 unlimited debug works

高 知

岡本 和樹 (3 年) 長澤 陸 (3 年)  
牧野 圭吾 (1 年) 谷澤 俊弘 (教員)

### 1. はじめに

今回の競技は、タイルポイントと領域ポイントの合計を競う競技である。タイルポイントか領域ポイントのどちらかを優先するかを unity の AI 機能で行動を決める。

### 2. 情報の視覚化

左図のような GUI を用いるので、現在の状況がプログラム使用者に視覚的に分かるようになる。

### 2. データの入力

マスの得点とエージェントの位置を QR コードで読み取り、unity に出力し左図のように表示する。

### 3. 行動決定

移動、タイル除去、停留を AI に処理させ、いずれかの行動を出力させ、次ターンは今ターンの相手の行動を入力し、AI に処理させ、行動を出力する。

|    |    |     |     |     |     |    |    |    |    |
|----|----|-----|-----|-----|-----|----|----|----|----|
| -8 | 8  | 8   | -14 | 4   | -12 | -9 | 11 | 13 | 14 |
| 2  | 4  | 15  | 3   | -12 | 13  | 14 | 15 | -7 | 9  |
| 1  | 11 | -13 | 0   | 7   | 14  | 6  | 5  | 7  | 4  |
| 8  | 13 | 12  | 6   | 3   | -3  | 9  | 6  | 14 | 15 |
| 3  | 11 | 5   | 7   | 8   | 7   | 0  | -6 | 14 | 5  |
| 6  | 12 | 4   | 8   | 2   | -10 | 10 | 9  | 14 | 4  |

図 1. プログラム実行時のイメージ

### 4. 開発環境

OS: Windows10

環境: unity/Visual studio 2017

言語: C#/python

## 52 何気ないプログラムが OS をきずつけた

旭 川

近江 雄哉 (4 年) 伊勢谷賢司 (4 年)  
斎藤 唯翔 (3 年) 嶋田 鉄兵 (教員)

### 1. はじめに

今回の陣取りゲームにおいての戦略は C++ のアルゴリズムによって探索する。また、GUI によって自陣と敵陣の情報を入力し、その状況での探索結果を表示する。

エージェントへの情報の伝達には、トランプを用いて行う。

### 2. アルゴリズム

エージェントの一人は中央から自陣を広げるように、もう一人は少し外側を囲みつつ相手の領域ポイントの取得を阻害するということを軸にアルゴリズムを作成する。

### 3. UI

図 1 のように、QR コードから得られたマップの情報を GUI 上にグリッドを表示し、マウスで操作することによってエージェントの状態を入力し、探索結果を表示する。

|            |    |    |     |    |     |     |    |    |    |
|------------|----|----|-----|----|-----|-----|----|----|----|
| MainWindow |    |    |     |    |     |     |    |    |    |
| メニュー       |    |    |     |    |     |     |    |    |    |
| 11         | 10 | 8  | -13 | 8  | 13  | -12 | 3  | -3 | -6 |
| 9          | 1  | 7  | -11 | 7  | 14  | 5   | 4  | 4  | 1  |
| 8          | 11 | 2  | -9  | 1  | 15  | 6   | 7  | 9  | 1  |
| 13         | 3  | 11 | 14  | 10 | -13 | 8   | -3 | 11 | 10 |
| 8          | 11 | 2  | -9  | 1  | 15  | 6   | 7  | 9  | 1  |
| 9          | 1  | 7  | -11 | 7  | 14  | 5   | 4  | 4  | 1  |
| 11         | 10 | 8  | -13 | 8  | 13  | -12 | 3  | -3 | -6 |

図 1 GUI の例

### 4. 開発環境

|          |                               |
|----------|-------------------------------|
| OS       | Windows 10                    |
| IDE      | Visual Studio 2017、Qt Creator |
| 言語・ライブラリ | C++、OpenSiv3D、Qt              |

## 53 SnailRush

広島商船

泉 瑞希 (3 年) 藤井 志帆 (4 年)  
福永 明那 (1 年) 岩切 裕哉 (教員)

### 1. はじめに

囲まれた内側の領域は、マス of 整数値を絶対値としてカウントする。このルールから領域ポイントを多く取得することを目指す。そのために、小さな領域を増やしながら進んでいき囲みを増やしていく。

### 2. プログラム概要

#### 2.1 序盤

まず、最短経路を選出する。次に全体の設計図を作る。片方が斜め方向の直線を描くなら、もう片方は垂直水平方向の直線を描くように設定する。

#### 2.2 中盤

その設計図通りに進行することを最優先する。もし、相手チームが先にタイルを置いた場合は先に他の経路に進む。

### 2.3 終盤

ある程度設計図通りの行動が終わったら、相手チームが先にタイルを置いたパネルの修復に向かう。

### 3. 開発環境

Java, OS: Windows10, intelliJidea



競技部門

## 55 人間の力

東京

柴田 紘希 (2 年) 永谷 黎 (4 年)  
東 佑輝 (2 年) 大前 佑斗 (教員)

### 1. アルゴリズム概要

このゲームは盤面の評価が非常に難しいので、我々はモンテカルロ木探索を用いた戦略を採用しています。ある手から試合終了までをランダムにプレイアウトし、勝率を計算することによって、評価関数の必要がないアルゴリズムを完成させました。

### 2. エージェントへの指示伝達方法

実際に練習環境を構築して実戦形式で練習を重ね UI の改善やアルゴリズム改善のフィードバックとしました。図 1 にあるように、次に司令塔が出すべきトランプを UI 上に表示させることによって、スムーズな情報伝達を可能にしています。

又、相手チームの暗号を傍受する事は 1 ターンの時間の短さから厳しいと判断しています。同様の理由から、相手チームも試合中に自チームの暗号を解読し利用することは難しいと考え、相手からの自チームの暗号の傍受に対する防御を捨て、早く正確に指示を伝達することに重きを置いた平易な暗号を採用しています。

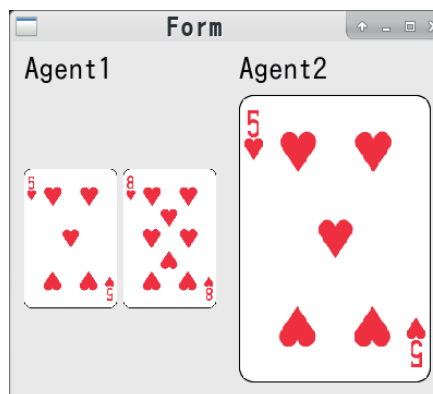


図 1

### 3. システム構成について

今大会のルールでは 1 ターンの行動選択にかけられる時間は少なく、その中で、もし PC を一台しか持ち込んでおらず、そのうえで不具合を起こしてしまうと、次の指示を出すのが極めて困難になってしまいます。そこで、複数のパソコンで通信してネットワークを構築し並列で計算させ、PC に不具合が生じることを想定し、フォールトトレランスを高めた構成にしています。

## 56 オールド AI-9685

福島

加藤 大貴 (4 年) 小野 和志 (4 年)  
奥山 拓人 (4 年) 小泉 康一 (教員)

### 1. 概要

QR コードをもとに初期盤面を読み込み、別タブにて現状の盤面より最善と判断したエージェントへの指示を提示、司令塔の指示の支持をする。

### 2. システム構成

#### 2.1 入力部

初期盤面の入力は QR の出力数列より生成、エネミーの移動はタッチによる入力で行う。自陣エージェントは最善解を自動で入力するが専らこれに依らず、手動での変更もできる。

#### 2.2 解析部

入力された盤面データより次の行動を点数により評価し、最善解として出力する。また、終了ターンが 60~120 と不明であるため 60 ターン以降はゲームの終了に備えて徐々に得点へ速攻性のある行動選択へ移行する。入力よりエネミーの行動が戦略テンプレートであると解析した場合は戦略の破壊を最優先に指示・行動する。

### 2.3 出力部

司令塔で戦況が確認しやすいよう、随時戦況バー及び色分けによりビジュアル化した盤面を図 1 のように更新・表示する。また、伝達段階でのミスを抑えるため指示は移動先ではなく実際に提示する A4 トランプのスイート及び数字を出力する。

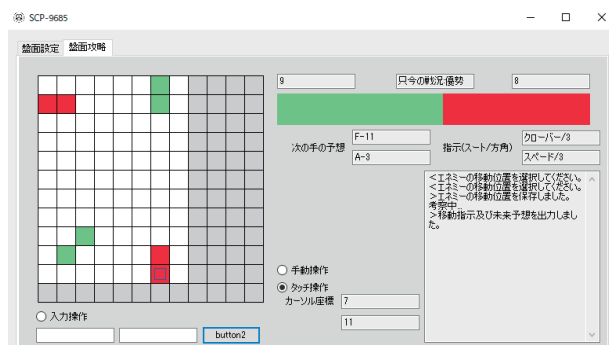


図 1. GUI イメージ

### 3. 開発環境

言語: Visual C++ Visual Basic

IDE: Visual Studio

## 57 計算じゃ解けない疑問は 手と手で探るの

久留米

菅野 暁 (5 年) 堤 幸太朗 (4 年)  
橋原 晃都 (3 年) 田中 諒 (教員)

### 1. 基本方針

チーム内で考察し、今回の競技を最適にプレイするには単純な一つの戦術ではなく相手の陣地を考慮した緻密な点数稼ぎが必要になるだろうと推測しました。それを人間がプログラム化したり調整したりすることは恐らく困難です。そこで基本方針としては、ディープラーニングを用いています。(ただし執筆当時はまだ開発中で、もう少しいかなかったら他の方針に変わっているかもしれません)

詳細については囲碁 AI の AlphaGo の手法を参考にし、規模の縮小や今回の競技に合わせた改変をしています。

### 2. 難しい部分

既存技術を応用するにあたって、囲碁などと違う今回の競技特有の性質に悩まされました。例えば囲碁と違い不完全情報 (同時手番に由来) であるため探索アルゴリズムも

ネットワーク構成も改変する必要がありました。また 1 チームが複数の駒 (エージェント) を操作すること、盤面サイズが可変であることなども考慮しなければいけませんでした。

最も大きな問題は、ディープラーニングのための強力な計算環境が無いことです。学校の PC 複数台で並列計算させてもまだ効率が悪く、種々の試行が数日単位でしか行えません。本番までの限られた日数を削りながら試行錯誤しています。

### 3. 最後に

夜寝る前には学校で孤独な学習をするネットワークを想い、昼目を覚ませばネットワークの様子を見にゆき、富める時も貧しき時も変わらぬ愛情を注ぎました。ネットワークがそれに応え、深遠な試合を見せてくれることを祈っています。

## 58 KKM

## 釧路

濱高 靖浩 (3年) 畑井 有人 (3年)  
堀内 琉郁 (3年) 天元 宏 (教員)

## 1. はじめに

OpenCvSharp を使用して web カメラから画像を取り込み、ZXing.Net を使用して QR コードをデコードする。そこから得られたフィールドの情報を基に探索を行う。

## 2. エージェントへの指示伝達方法

用意されているトランプを用い、一人に対して 2 枚提示する。1 枚目はトランプの絵柄にそれぞれ移動、タイル除去、停留を割り当てたものを提示する。また、2 枚目は数字に行動する方向を割り当てたものを提示する。

## 3. アルゴリズム

探索には A\*探索を基にしたアルゴリズムを用いる。

評価値は各タイルのスコアを基準とする。ただし、タイルポイントの減点を防ぐために負の数のタイルの評価値は 0 とする。

また、評価関数は経過したターンやスコアの優劣によっ

て変える。ターン数を半分に分けて前半はフィールド外側のタイルの評価値を高くし、後半は中央のタイルかつ相手の占領タイルの評価値を高くする。これにより、中盤までにタイルポイントが優勢な状況を作る。

すべてのタイルを評価し、最も評価値が高いタイルを解とする。1 ターンごとの制限時間が短いため、探索が長引いた場合は中断してその時点の解を導出する。

## 4. GUI

相手エージェントの行動の入力に加えて、フィールドの全体像、現時点での両者の合計スコア、各エージェントの位置、タイルの占領状況、探索の解を表示する。

## 5. 開発環境

OS : windows10 言語 : Ruby, C#

エディタ : VisualStudio 2017, sublime text

ライブラリ : OpenCvSharp, ZXing.Net, FXRuby

## 59 巡りマス人力の方が強くない？

## 大阪府立

吉田 凌河 (3年) 尾藤祐一郎 (3年)  
旨森 貴一 (3年) 窪田 哲也 (教員)

## 1. はじめに

今回の競技部門はマス目に区切られたフィールドでいかに多くの陣地を占領できるかを競う陣取りゲームである。1 ターンで四人のエージェントの移動方向とタイルをはがすかどうかを考えなければならないため効率的な探索方法を考える。

## 2. ソルバーのアルゴリズム

基本は全探索だが、それだと時間内に良い手が見つからないため、良いスコアが得られそうにない手は枝刈りをして探索する局面を減らす。

## 3. 特徴・工夫

マウスでカメラの移動・拡大・縮小ができる。ソルバーの考える最善手を 1 秒ごとに更新して Visualizer に表示することによってどの行動をすべきなのかが瞬時に分かるようになる。



## 4. 開発環境

Python, C#, OpenCV, Unity

## 1. はじめに

今回の競技はターン制の陣取りゲームのため、ゲームルール処理サーバーを用意し、ゲームの盤面データを管理したうえで、そのサーバーから得られる盤面情報を基にしてソルバーが次の手として最適な手を選ぶという方式をとる。また、相手チームの操作などのサーバーの情報更新とソルバーの解の表示のために、UI を用意する。

## 2. 各ユニットについて

### 2.1 ゲームルール処理サーバー

盤面の情報などの各種情報の保持、管理を独立して行い、必要に応じてソルバーや UI に対して情報を供給する。

基本的に盤面の情報は UI から入力されてきたものを用いる。

### 2.2 操作 UI

各ターンにおける相手チームのコマの移動状況入力や、

ゲームが始まった時点での盤面の状態を QR コードから読み取るなどの処理を行い、サーバーに情報を伝達する。

### 2.3 ソルバー

DQN で、タイルの点数に分けてチャンネルを作るなどして学習をさせたソルバーを用いてその場の状況に最も合った手を選択する。

## 3. エージェントへの指示伝達について

エージェントへの指示伝達はカードの図柄と指示を一对一対応させて行う。具体的にはトランプのある二つの数字に移動とマス除外の二つの処理を割り当て、それ以外の数字に行動を起こす方向(八近傍と移動しない)を割り当て、エージェントに二つのカードを提示することでエージェントの行動を指示する。また、各エージェントに対してトランプの図柄一つを対応させることによって指示対象のエージェントを特定する。

## 1. Introduction

Our system will show the moves on its screen and do computation on previous moves then try to find the best move.

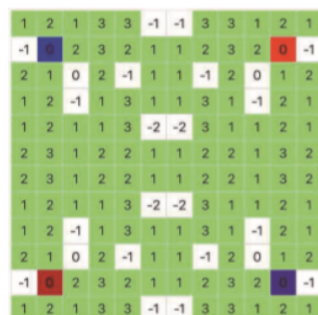


Figure 1. Software screenshot

The system is capable of doing following actions:

- Draw board on the screen
- Make changes on the board after opponent's move
- Compute its next move
- Make changes on the board after its move

Currently, the system is under development. We will improve our algorithm further.

## 2. Programming language

The system is written in Javascript. Because Javascript is the one of the modern, powerful languages and it is easy to make web interface for our system.

**I. Introduction**

Our program is a Web application, which is divided into two parts: Client-side program and Server-side program. Client-side program includes a user interface which allows the Tower to update the game's state easily, while Server-side program provides the Tower suggestions of next moving steps.

**II. Client-side program**

We used HTML, CSS and JavaScript to write the Client-side program. It provides the Tower with several tasks such as import of competition field data, updating two teams' move, calculating 2 teams' score and competition field data backup for recovering from program failure. Our Client-side program can help the Tower to grasp 2 teams' situation in order to decide what strategy should be used.

**III. Server-side program**

The Server Application was written in C++ and is run as a service on Web server. Whenever the status of two teams are updated by Client-side program, the Server-side program will calculate the best solution for the next moving steps. In order to do that, we use Minimax algorithm and Alpha - Beta pruning algorithm. The descriptions of the algorithms are as follows.

**3.1 Minimax algorithm**

Minimax is a kind of backtracking algorithms that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. Thus, we use this algorithm to build a game tree which leads to all states available after  $n$  steps from current state. Our program then searches through the tree to find out the best solution which can give the highest profit.

However, there are too many states that need to be searched in the tree. Hence, it takes our program a lot of time to process and exhausts computer memory. To reduce the calculation cost and memory, we use Alpha - Beta pruning algorithm.

**3.2 Alpha - Beta pruning algorithm**

Alpha - Beta pruning is a searching algorithm that try to decrease the number of nodes that need to be evaluated by the minimax algorithm in its game tree. It cuts off branches in the game tree which need not searching because there already exists a better node available. By using this algorithm, our program does not only search faster but also goes into deeper levels of the game tree. Therefore, our strategy is able to find the best solution for next moving steps within given time.

**1. Introduction**

The objective of the system is to optimize the total points of tile and area of the "Surround the Squares" problem. Given QR codes for the points of the squares and locations of the Agents, the Control Towers, at each stage, are in the position of knowing which square to move to. The decision made for each turn has its corresponding rewards and risks. Therefore, maximizing rewards by accumulating as many points as possible and minimizing risks by preventing the opponent from surrounding our squares are desirable.

**2. Methods and Functions****2.1 Reading QR Codes**

Using a camera, QR codes will be read and the points for the squares and the locations of the Agents are known.

**2.2 Algorithms**

Firstly, inputs are copied to a text file. Later, all squares and their corresponding points are gathered in a graph for better visualization.

At an Agent's current location, there will be eight possible choices. The program employs a greedy approach. At each turn, a locally optimal choice *i.e.* the un-visited square with the most point, is being selected.

**2.3 Implementation**

For the implementation, we use Python.

**2.4 Output**

The program calculates the expected points to be accumulated.

## 1. Introduction

This year, in the Competition Section, we will conduct a type of a territory game to fight over which team can take more squares on a field divided up into grids.

## 2. Overview of the developed program

### 2.1 Introduction of System

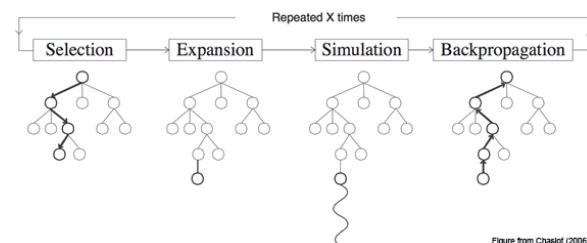
We had developed a set of tools to help us to determine which strategy we are going to use. Like the “Score Map Tool”, it can be used to display the score in different color, If the score of the square is tend to +16, the color of background will tend to light green, and vice versa.

### 2.2 Overview of the algorithm

We had added artificial intelligence (AI) to help us for making

optimal decisions.

Monte Carlo Tree Search (MCTS) combines the generality of random simulation with the precision of tree search.



**Selection** - Starting at root node  $R$ , recursively select optimal child nodes (explained below) until a leaf node  $L$  is reached.

**Expansion** - If  $L$  is not a terminal node (i.e. it does not end the game) then create one or more child nodes and select one  $C$ .

**Simulation** - Run a simulated playout until a result is achieved.

**Backpropagation** - Update the current move sequence with the simulation result.