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。学生はプレイヤーとなってフィールド上を移動する複数のエージェントの行動を決定します。試合はターン制で進行し、1 ターンごとに両チームのエージェントが同時に行動します。プレイヤーは、決められた時間以内に全てのエージェントの行動をサーバーへと送信しなければなりません。一定のターン数が経過した時点で、エージェントが占有した陣地のポイントによって、チームの勝敗が決まります。

● フィールド

フィールドは最大で縦 20×横 20＝合計 400 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。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、線対称に配置されます。

● 試合の進行

試合は、公開フィールド戦と非公開フィールド戦の 2 回の対戦を行います。公開フィールド戦で使用するフィールドは試合開始前に連絡し、非公開フィールド戦のフィールド情報は非公開フィールド戦の対戦開始と同時に回答システムから取得することが可能になります。

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。

● 試合形式

1 チーム対 1 チームの試合を同時に 3 試合、つまり、チーム A がチーム B、C、D と同時に試合をする可能性があります。ファーストステージ、セカンドステージ、ファイナルステージに分かれており、ファーストステージとセカンドステージは 3 または 4 チームによるリーグ戦となります。また、ファーストステージで敗退したチームの敗者復活予備戦、敗者復活戦も 3 または 4 チームによるリーグ戦で行われ、敗者復活戦のリーグ 1 位のチームがセカンドステージに進みます。ファイナルステージは、8 チームによるトーナメント戦を行います。

● 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. タイルポイントと領域ポイントの合計ポイントが大きい方のチームが勝利します。
2. 合計ポイントが等しい場合、タイルポイントが大きい方のチームが勝利します。
3. 合計ポイントとタイルポイントが等しい場合、トランプなどで勝敗を決めるか引き分けとします。

競技部門の組合せ

ファーストステージ

リーグ		チーム	リーグ		チーム	リーグ		チーム	リーグ		チーム
1－A	a	都立(荒川)	1－B	a	香川(詫間)	1－C	a	長野	1－D	a	仙台(広瀬)
	b	呉		b	福島		b	奈良		b	北九州
	c	熊本(熊本)		c	石川		c	弓削商船		c	米子
	d	釧路		d	岐阜		d	国際		d	
1－E	a	大島商船	1－F	a	津山	1－G	a	舞鶴	1－H	a	長岡
	b	和歌山		b	松江		b	富山(射水)		b	鹿児島
	c	鈴鹿		c	東京		c	高知		c	八戸
	d	阿南		d	秋田		d	大阪府大		d	
1－I	a	豊田	1－J	a	香川(高松)	1－K	a	苫小牧	1－L	a	一関
	b	旭川		b	鶴岡		b	小山		b	鳥羽商船
	c	新居浜		c	広島商船		c	都城		c	沖縄
	d	茨城		d	久留米		d	有明		d	
1－M	a	都立(品川)	1－N	a	サレジオ	1－O	a	熊本(八代)			
	b	宇部		b	徳山		b	佐世保			
	c	大分		c	仙台(名取)		c	福井			
	d	木更津		d	神戸市立		d	沼津			

※ 各リーグで1位と2位のチームがセカンドステージに進出、3位以下のチームは敗者復活予備選に

敗者復活予備戦

リーグ		チーム	リーグ		チーム	リーグ		チーム	リーグ		チーム
R1－A	a	1-A-4	R1－B	a	1-A-3	R1－C	a	1-E-4	R1－D	a	1-E-3
	b	1-B-3		b	1-B-4		b	1-F-3		b	1-F-4
	c	1-C-4		c	1-C-3		c	1-G-4		c	1-G-3
	d	1-D-3		d			d	1-H-3		d	
R1－E	a	1-I-4	R1－F	a	1-I-3	R1－G	a	1-M-4	R1－H	a	1-M-3
	b	1-J-3		b	1-J-4		b	1-N-3		b	1-N-4
	c	1-K-4		c	1-K-3		c	1-O-4		c	1-O-3
	d	1-L-3		d			d			d	

※ 各リーグで1位のチームが敗者復活決定戦に進出

敗者復活決定戦

リーグ		チーム	リーグ		チーム
R2－A	a	R1-A-1	R2－B	a	R1-B-1
	b	R1-C-1		b	R1-D-1
	c	R1-E-1		c	R1-F-1
	d	R1-G-1		d	R1-H-1

※ 各リーグで1位のチームがセカンドステージに進出

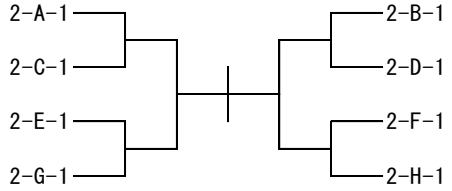
セカンドステージ

リーグ		チーム	リーグ		チーム	リーグ		チーム	リーグ		チーム
2－A	a	1-A-1	2－B	a	1-A-2	2－C	a	1-E-1	2－D	a	1-E-2
	b	1-B-2		b	1-B-1		b	1-F-2		b	1-F-1
	c	1-C-1		c	1-C-2		c	1-G-1		c	1-G-2
	d	1-D-2		d	1-D-1		d	1-H-2		d	1-H-1
2－E	a	1-I-1	2－F	a	1-I-2	2－G	a	1-M-1	2－H	a	1-M-2
	b	1-J-2		b	1-J-1		b	1-N-2		b	1-N-1
	c	1-K-1		c	1-K-2		c	1-O-1		c	1-O-2
	d	1-L-2		d	1-L-1		d	R2-A-1		d	R2-B-1

※ 各リーグで1位のチームがファイナルステージに進出

リーグ戦の順位決定方法
リーグ戦では合計ポイントとタイルポイントが等しい場合は引き分けとし、以下の優先順位で決定する。
１．勝率が高いチームを上位とする（勝ち数／全試合数）
２．当該チーム間の試合の勝率が高いチームを上位とする
２－２．さらに当該チーム間の試合の勝率が高いチームを上位とする
３．当該チーム間の試合の得失点率の高いチームを上位とする
４．当該リーグ戦全試合の得失点率の高いチームを上位とする
５．主催者が指定する方法で上位とする

ファイナルステージ



※ トーナメント戦では合計ポイントとタイルポイントが同じ場合、引き分けとはしない。
その場合の勝敗の決定方法は本選当日に告知する

1 クレマスカ?¿

呉

水本 直希（4年） 長橋 朋也（4年）
吉岡明佑武（3年） 藤井 敏則（教員）

1. はじめに

前回と競技内容はほとんど同じであるが、前回用いたディープラーニングによる推論ではなく、ビームサーチを行うことで次の手を算出するシステムを開発した。

2. ステップ毎の進行

(1) 作戦ステップ

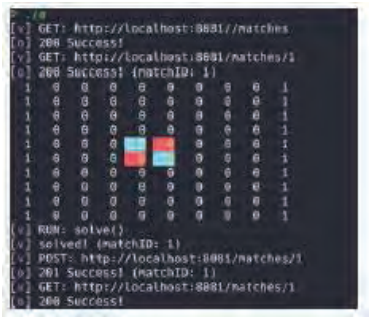
作戦ステップではエージェントの行動を決定し、結果を送信する必要がある。複数のパソコンで分担して処理を行うことも考えたが、今回は並列処理を利用して一つのパソコンで計算を行う。公開フィールドか非公開フィールドによってソルバの種類を変更し推論を行う。

(2) 遷移ステップ

遷移ステップでは特にすることはない。後述する CLI の出力を人力で見守る。

3. 推論ソフトウェアについて

今回の競技では ほとんどすべての操作が人間を介することなく行えるため、グラフィカルなインターフェースは必要ないと考えた。そこでシンプルな CUI を採用し、開発の工数を少なくした。



4. 開発環境 ・ ライブラリ等

・WSL(Ubuntu) 、 g

・picojson 、 cmdline

2 Herbivores

国 際

勝又 舜介（2年） 青木 心路（2年）
深山 寧皇（2年） 伊藤 周（教員）

1. はじめに

我々のチームは、ポイントを稼ぐエージェントと敵の妨害及び、味方陣地の防衛を同じプログラムで行うのは無理があると考えた。そこで、我々はエージェントを二つの方法で動かすことにした。一つ目はプログラムのみで行動を決定するエージェントで、二つ目は我々選手が行動を決定するエージェントである。

2. プログラムで動かすエージェントについて

プログラムで行動を決定するエージェントの役割は、領域ポイントを獲得し点数を稼ぐことである。そこで、我々はフィールドのタイルのポイントと、そのタイルをとることで得ることができる可能性のある領域ポイントを加味した「価値マップ」というものを作成する。エージェントの行動決定方法はダイクストラ法を使用し、価値マップの情報を基にした仮のゴールを定め経路を作る。経路毎に得られるポイントの合計と必要なターン数から期待値を計算し、最も効率の良い経路を発見する。仮のゴールに着いたら、初期位置を次のゴールとして同様の手法で移動する。その際、相手の陣地と味方の陣地を加味した状態で価値マップを再度作成し、同じ経路を通らないように初期位置に戻る。

3. 選手が動かすエージェントについて

選手が動かすエージェントの役割は敵の妨害及び、味方陣地の防衛である。そこで私たちは先述した価値マップに相手のエージェント情報を盛り込んだ「ナビマップ」を毎ターン作成する。その情報と自分たちの経験などを基に選手はエージェントの行動を決定する。

4. 開発環境

Python, Visual Studio Code, Windows power shell,

3 アルティメット・マジシャンズ

サレジオ

小坂 優介（3年） 香月 優太（2年）
小島　 実（2年） 宇都木修一（教員）

1. はじめに

開発するシステムは、「サーバークライアント通信」、「探索」、「GUI」の3つの要素から構成される。以下にそれぞれの説明を記す。

2. サーバークライアント通信

競技サーバーからターン開始時にフィールド情報を取得することと、探索結果からエージェント行動情報を送信することを行う。

3. 探索

探索は公開フィールドと非公開フィールドで異なるアルゴリズムを使用する。

公開フィールドでは事前に公開されるフィールド情報を使用し、機械学習を行う。機械学習で生成したデータを使用し、最も勝てる確率が高い手を選択していく。AlphaZeroというアルゴリズムを使用する。

非公開フィールドではUCTというアルゴリズムを使用する。効率良く次の手の候補を選択し、試行を繰り返し、最も勝てる確率が高い手を選択するものである。

4. GUI

画面上にフィールドの状態を描画する。図1に表示例を示す。

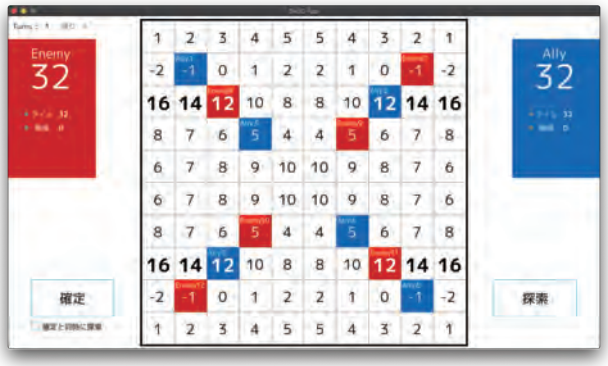


図1. GUI

探索が終了すると探索結果も画面上に表示される。このとき探索が失敗した場合を考え、手動でエージェントの行動入力も行えるようになっている。

5. 開発環境

IDE：Xcode、言語：C++、ライブラリ：OpenSiv3D

4 八門金鎖の陣

新居浜

松本 楓真（4年） 濱岡　 空（4年）
渡部 一真（4年） 占部 弘治（教員）

1. 『八門金鎖の陣』とは

三国志演義において曹仁が繰り出した高度な陣形のことで、攻撃にも防御にも使うことができる。どんな状況でも臨機応変に対応できるシステムになるようにこの名前を付けた。

2. UI について

ヒューマンエラー等に迅速に対応しなければならないため、直感的な操作で分かりやすい GUI を作成する。フィールドの状況やエージェントの位置を分かりやすく画像で表示し、移動指示をマウスやタッチパネルで簡単に行えるようにし、素早く指示を出せるようにする。

3. アルゴリズムについて

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。そこで、自チームのポイントの最大化を狙い、司令塔からの正確な指示ができるように設計を考える。1 ターン当たりの時間が短く、探索にかけることができる時間が少ないため、ビームサーチを用いて探索を行う。フィールド上で得られるデータや敵エージェントとの距離、タイルの状況などから評価関数を算出し、評価関数が最も大きいものを最適な解とすることで計算を単純化し毎ターンその結果から指示を出すことにする。

4. 開発環境

言語：C#, Python

環境：VisualStudio 2019

OS：Windows8.1/10

5 パズルソルバー2

鳥羽商船
弓削 隼大（2年）林　　蓮（2年）
小畑 慧吾（1年）土田 隼之（教員）

1. はじめに

今年の競技部門のルールは、前回に引き続き陣取りゲームである。我々のチームは、モンテカルロ木探索による、勝率が高い経路を優先的に選ぶ方法を採用した。

2. 公開フィールドでの戦略アルゴリズム

探索アルゴリズムとしてモンテカルロ木探索を使用し、ランダムに勝率の高い手を探索する。この際、探索範囲が広くなるとその分計算時間が大幅に増加する。そこで、ディープラーニングを用いた評価関数でそれぞれの手を評価して探索範囲を限定する。これにより、モンテカルロ木探索のみを用いて最大ターン数まで探索する方法と比べ、勝率を高めるための計算時間を削減できる。

3. 非公開フィールドでの戦略アルゴリズム

非公開フィールドではディープラーニングを行う時間はなく、ディープラーニングにより生成した評価関数による探索評価は不可能である。また、全ての手について探索する全探索は計算量が大幅に増加するため現実的ではない。そのため、評価関数を約２０～３０ターンの探索範囲での得点数や得点見込み点としてモンテカルロ木探索を用いてランダムに勝率の高い手を探索し、最も勝率の高かった手を返す手法を用いる。

4. 開発環境

言語：C, Python, Scala

エディタ・IDE：VisualStudioCode, VisualStudio

OS：Windows10, MacOS, Linux(LinuxMint)

6 クウェウエクウェウエ

香　川
（高松）
高木　綸（3年）三浦　翔（3年）
竹内 歩夢（3年）北村 大地（教員）

1. はじめに

今回の競技部門では、マス目に区切られたフィールド上で多くの陣地を占領できるかを競う陣取りゲームである。システムの要素として「システムの動作概要」、「アルゴリズムの概要」とした。

2. システムの概要

フィールドのタイル配置やエージェント配置等の入力情報に基づいて有利な次の動作を算出し、回答システムに送る動作である。有利な次の動作とはアルゴリズムを用い自陣のエージェントの近辺にある得点の高いタイルを数個先までそれぞれ探索し、一番得点が高くなる動作をするというものである。さらに相手のエージェントの行動を見ながら適宜それに合わせた行動をとる。

3. アルゴリズムの概要

各ノードの数手先を探索し、評価値の高いノードを抽出して探索を繰り返すビームサーチというアルゴリズムを使用する。このビームサーチでは評価値の高いノードだけを探索することにより時間短縮ができるためより深くのノードを探索できる可能性があるという長所がある。

4. 開発環境

言語:C/C++,Python

環境:Visual Studio Code, Visual Studio 2019

外部ライブラリ:

picojson

「https://github.com/kazuho/picojson」

OpenSiV3D

「https://github.com/Siv3D/OpenSiv3D」

7 チーム名をあらん限りの声で叫んでいます（嘘、叫ばないで）。

一　関
菅原　光（5年）松原 優太（4年）
市村 大陸（2年）若嶋振一郎（教員）

1. はじめに

今回の競技はエージェントの人数やフィールドの大きさが変動し、どちらも最大の状態では計算に膨大な時間がかかると考えられるため、探索を行う際には適宜枝刈りをしながら計算を行っていく、また、公開フィールドと非公開フィールドの 2 種類のフィールドそれぞれについて個別の戦略を取ることが可能である。

2. フィールド情報の取得

C++ REST SDK を使用して HTTP リクエストを行いフィールド情報の取得を行う。取得した情報は Json.NET や picojson を使用してデシリアライズし変数などに格納することで、プログラムで扱えるようにする。

3. 公開フィールド

強化学習の手法の一つである R2D2 を用いて、公開フィールドのおける最適解に近い行動を見つけ出す AI を育成する。AI を用いてその盤面における行動を決定し、各エージェントに指示を出す。

4. 非公開フィールド

エージェントの行動後の盤面をノードとしたゲーム木を展開する。エージェントの行動先は得点が高いマス数個+ランダムなマス数個に絞ることで計算時間の削減等を行っている。展開した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。ビームサーチの評価関数ではスコア差やエージェントの位置関係などを考慮して評価を行う。

5. 開発環境

【OS】Windows10
【言語】C++/C#/Python
【エディタ】Visual Studio 2019
【ライブラリ】Boost/Prism/Json.NET/picojson/
C++ REST SDK

8 そうだ、全探索しよう

福　島
菅野 聖真（4年）小宅 大地（4年）
園部 航生（4年）小泉 康一（教員）

1. はじめに

今回作成したアプリケーションについて、「探索アルゴリズム部」「描画部」「通信部」に分けて以下に示す。

2. 探索アルゴリズムについて

今回作成したアプリケーションでは、主にモンテカルロ木探索を用いて探索を行う。しかし、エージェント数が多くなると 1 ターンに動かすことのできる手数が膨大になることから、縦横 4 方向への探索を放棄し、斜め 4 方向のみ探索することにした。これにより、探索コストを大幅に削減し、より深くまで探索できるようにする。

タイル除去行動については、各エージェントに対し 8 方向に判定を行い、それにより相手のポイントが 6 ポイント以上減少する場合のみその行動を優先してとるようにする。

また、エージェントの移動データを 2×(エージェント数)ビットの整数で管理することで、ビットマスクにより高速に各エージェントの移動データを取り出すことが可能になっている。

3. UI の作成について

UI の作成には、C++のライブラリである openFrameWorks を利用した。

4. サーバとの通信について

コマンドプロンプトで実行できる curl コマンドをそのまま C++のターミナルから実行することで、高速に通信できるようにした。

5. 開発環境

使用言語 ： C++
IDE ： Visual Studio 2017

9 俺たちはソルバーになれない

熊　本（熊本）
末永 和大（5年）白川 嵩大（5年）
扇塚 和希（4年）孫　寧平（教員）

1. 概要

今回の競技部門は、特に公開フィールドと非公開フィールドの2つが存在し、それぞれに有効な解を短時間に得ることが鍵となる。これには高速で解を求めるプログラムが必要である。しかし、解を求めたところで最終的にはサーバへ送信する手順が発生し、短時間でありながら多少複雑な操作を要求される。また、明らかにおかしい動きをするエージェントが存在する場合も考えられる。

これらに対応するため、汎用型演算装置†NINGEN†を利用することにより不測の事態に対応する事を決めた。

2. アプローチ

2.1 公開フィールド

公開されているフィールドに対して機械学習を用いることで、それぞれの盤面に有効なニューラルネットを構築して対応する。

2.2 非公開フィールド

次の手の決定を複数の評価関数を作成し、これらの評価関数による評価に対し、遺伝的アルゴリズムを用いて最適化する。最適化された評価関数を用いて、複数の探索方法を試し、勝率の高い結果を採用する。エージェントの数も多いことから枝切りなど計算量削減を図る。

2.3 操作方法

不測の事態に対応する汎用型演算装置†NINGEN†は、即時のプログラム操作を要求される。これには、フィールド更新、移動先を瞬時に求め、サーバへ送信するGUIが必要であり、移動先は修正が可能な、どのような環境でも汎用性が高い†NINGEN†の判断が反映できる機能を用意した。

3. 使用環境

Windows10, Mac OS X, Visual Studio 2019, XCode
C++, C#, Python, JS, Siv3D, Node.js

10 なんで勝てたか明日までに考えときます

松　江
鈴木　豪（4年）青木 蓮樹（4年）
松江 理恵（2年）橋本　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，ゲームを考察するための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

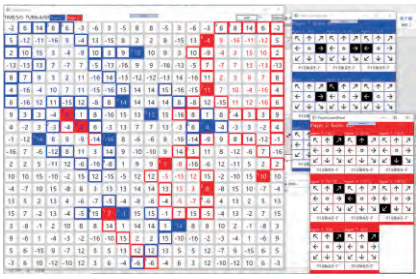
基本的にゲーム木を縦型探索（深さ優先探索）している．ミニマックス法や，明らかに不利になりそうな手の枝刈りによって，探索空間を減らしていることが特徴である．

2.2 評価関数

ゲームの得点だけでなく，塗った領域の形，マスとエージェントの距離なども考慮して評価関数を作った．また，公開フィールドの場合は，パラメータを遺伝的アルゴリズムによって最適化することで特別に強くしている．

3. インターフェースプログラム

開発段階で思考プログラムの動作を確認するために，ゲームの進行状況と思考プログラムの思考状況を表示するプログラムを作成した．さらに，本番でも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，思考プログラムがサーバに送信する手を変更する機能を持つ．



図．インターフェースプログラムの画面

4. 開発環境

言語　：C#, XAML
IDE　：Visual Studio 2019
OS　：Windows 10

11 留年から高専生を守る党

神戸市立
入本 聖也（1年）石原　実（4年）
上村　裕（4年）朝倉 義裕（教員）

1. はじめに

前回（去年）の方法では、モンテカルロ木探索の方法で各局面に対する手の評価を計算し、行動を決めるというものでした。この方法では評価の精度を上げるほど膨大な計算量を消費します。ここにニューラルネットワークの導入という最近の将棋や囲碁などの AI が採用している方法をとります。

2. ニューラルネットワークの導入

今回は２種類のニューラルネットワークを使用します。１つは局面の評価を計算するもの（以下価値ネットワーク）、もう一つはエージェントの行動の評価を計算するもの（以下方策ネットワーク）です。これらを上記のモンテカルロ木探索ベースのシステムに取り入れます。

3. 価値ネットワーク

価値ネットワークの出力値は上記のモンテカルロ木探索の Rollout という処理を繰り返した結果による評価値の代わりに置き換えられます。これで計算量が大幅に減少します。

4. 方策ネットワーク

方策ネットワークの評価は各エージェントの行動のフィルタリングに使われます。１エージェントの動きが１７通りありますから、上位のいくつかの行動のみを採用すれば計算量が大幅に減少します。

今回我々は、上の 2 種類の CNN を使用し計算量を大幅に削減させたモンテカルロ木探索ベースの探索法を採用しました。

12 八代の三銃士

熊　本
（八代）
古川 照英（3年）船越 大輔（3年）
山本 健貴（1年）小島 俊輔（教員）

1. 探索

まず全探索を用いる。全探索の深さは制限時間やアルゴリズムの複雑さを基に決定する。全探索の結果から、あらかじめ決めておいた評価値に満たないものは枝刈りし、モンテカルロ木探索を用いてより最適に近い解を見つけ出す。

1.1 全探索アルゴリズム

全探索は計算量が非常に多いため、探索に時間がかかる。短時間に効率よく探索するためにスレッドを用いて、同時に探索を行う。

1.2 モンテカルロ木探索

ランダムに一つ一つの手を選んでいき、何手か探索を行う。また時間を効率よく使うために全探索と同じようにスレッドを用い、さらに探索中の手に良い結果の望みがない場合、枝刈りを行う。

1.3 枝刈り

シミュレーション等を行い、どのような方法でタイルを置いても得点において解が期待できない場合の基準値を求める。その基準値を用いて枝刈りを行う。

2. ユーザインターフェース

今回、競技で使用する JSON、TCP/IP、UI のライブラリを様々な機能において processing が優れていたため、processing を用いてシステムの開発を行った。

今回のルールでは、人間による操作はほとんど必要ないと思われる。そのため、操作用のボタン等は最小限に抑える。だが、通信が不能になった場合や探索のエラーが発生する場合に対応できるよう、手動でデータの送受信の管理、探索のやり直しや途中までの結果を基に行動を決定できるようにする。

13 衣染めまく欲しけども 着てにほはばか 人の知るべき

香　川（詫間）
漆原 和輝（3年） 杉浦 翔真（4年）
若山 稜太（4年） 宮崎 貴大（教員）

1. はじめに

今回は、強化学習を用いてより高得点をとれるようなシステムを構築した。

2. システム概要

2.1 サーバーへのアクセスについて

送信は処理が終了した時点で自動的に行えるようにする。また、1ターンの時間を超えてしまわないように制限時間を設け作戦ステップ内で処理を終わらせ出力ができるようにする。

遷移ステップの明確な時間が分からないため相手の行動を取得するタイミングは人が計る。

2.2 行動決定について

Q学習を用いて学習を行い評価の良かった行動を行う。しかし、非公開フィールドではフィールド情報やエージェント数が分からないためランダムでフィールドを生成し学習を行うのは非効率である。そこで、フィールドの大きさやターン数を規定よりも減らし学習を行わせ、そこから徐々に増やしていくことで学習の効率化を目指す。

公開フィールドでは、フィールドごとに特化させた AI を試合で用いる。

3. GUI について

行動の取得を手動で行うため GUI のボタンなどを用いて簡単に取得できるようにする。また、フィールドの大きさの変化が大きいためマスの大きさを変化させ対応する。

4. 開発環境

OS ： Windows10

IDE ： PyCharm Community Edition 2019

言語 ： Python3

ライブラリ ： Tkinter , TensorFlow

14 紫陽花になったなめくじのように

久留米
樋口 陽介（3年） 稲田 雄大（2年）
佐伯 勇太（2年） 田中　諒（教員）

1. 概要

今回の競技は、公開フィールドと非公開フィールドの二種類のフィールドで対戦を行う。そのため、公開フィールドにおいては強化学習を行い、非公開フィールドにおいては探索を中心としたアルゴリズムを採用した。
また、入力、出力ともにサーバーとの通信で行った。

2. 公開フィールドにおける戦略

今回は、公開フィールドが事前に配布されるため、強化学習を行い、それぞれのフィールドに適した評価関数の構築を行った。

3. 非公開フィールドにおける戦略

基本的には、モンテカルロ木探索を応用した探索を行った。

今回は、エージェントの数が合計で最大 16 体と多く、探索空間が大きいため、探索を十分な回数行うことが難しい。そのため、探索の質を向上させるという判断を行った。
そこで、盤面の状態から、各エージェントの行動を評価する比較的計算コストの小さい評価関数を用いてプレイアウトを行い、各行動の価値を更新し、最も価値の高い行動を選択するというアルゴリズムを実装した。

4. サーバー通信

暗号化などは行われていないことから、通信の確実性が主となるため、通信の成功を確認する処理を行った。

5. 開発環境

MacOS, C++

15 大島商船ソフトボール同好会

大島商船
熊野 雄介（4年） 領家 直哉（5年）
岡部 蒼太（専1年） 北風 裕教（教員）

1. はじめに

本システムは、モンテカルロ木探索をもとにエージェントの行動を決定する。序盤は攻撃型のアルゴリズムで試合を進めていき、試合展開に応じてプレイヤーが戦略アルゴリズムの切り換えを行う。

2. 戦略アルゴリズム

エージェントには、“攻撃型”、“防御型”、“妨害型”の3種類のアルゴリズムを用意する。

2.1 攻撃型

モンテカルロ木探索を用いて点数の高いタイルや閉領域を確保する.

2.2 防御型

すでに味方エージェントが確保している高得点のタイルや閉領域に対して、相手エージェントが容易にタイル除去を行うことを防ぐ。

2.3 妨害型

相手の閉領域に対してタイル除去行動を行うことによって相手チームの得点を減らす。

3. 非公開フィールドでの経路探索

非公開フィールドでは、1ターンの間にモンテカルロ探索を行うことは困難であるため、貪欲法で行動を選択し、特定の回数を終えた時点で貪欲法からモンテカルロ木探索に切り替えを行う。また、非公開フィールドでのモンテカルロ木探索の精度は低下するため、公開フィールドよりも早いタイミングで味方エージェントのアルゴリズムの切り替えを駆使して試合を進めていく。

16 平成最後の時うさぎ

福　井
加藤 拓也（4年） 青山 椎耶（4年）
福田 拓水（4年） 村田 知也（教員）

1. システム概要

初めに、配布されたサーバーから情報を読み取る。読み取ったデータから適切な移動先を求め画面上に表示する。それをもとに最終的な決定を司令塔が行う。

2. アルゴリズム

2.1 序盤の移動

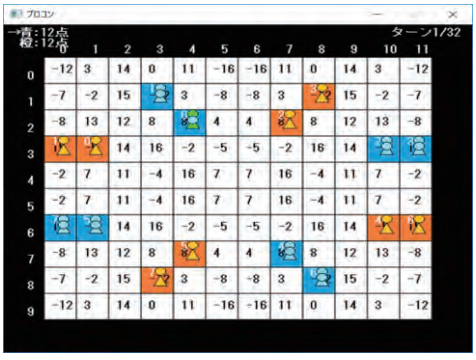
序盤は幅優先探索を用いて、2手先までの行動の中でより得点が取れる方法を求めて画面に表示し、基本的にそれ通りに移動していく。

2.2 終盤の移動

アルゴリズムは序盤の移動と同じ幅優先探索を用いるが、相手チームの移動を妨害することで相手チームの領域ポイントを減らせる場面などがあれば、司令塔の判断で移動先を最終決定する。

3. GUI

図のようなフィールドの状況が直感的にわかるようなGUIを用いる



4. 開発環境

OS：Windows10

使用言語:C言語,Python

IDE:Visual Studio 2019

17 米子のGON

米　子
権代 大翔（3年）佐野　楓（3年）
吉塚 創也（3年）徳光 政弘（教員）

1. はじめに

フィールド上での重要な地点であるパネル得点の高い地点または効率よく領域ポイントを得るために重要になるキーポイントを探しどの地点を重点的に探索していくかを決めることが重要になる。

2. アルゴリズム

2.1 公開フィールド

公開フィールドでの戦略アルゴリズムは、深さ制限探索を用いる。深さ優先系の探索を用いることによってエージェントが基本的に優先していく戦略を優先させながら探索させる。深さ制限探索に使用する主な制限としては、何手先だけを読むかというターン制限、特定の決まった条件下での定石が使用できるか判断する定石制限を主な制限として使用していきプログラムが効率よく解答を得られるようにする。

2.2 非公開フィールド

非公開フィールドでの戦略アルゴリズムは計算時間等の観点から公開フィールドと同様、深さ制限探索を用いる。次に、探索の手順を述べる。まず高得点を少ない手数で獲得するために大きな領域の形成を考える。しかし、大きな領域は相手に破壊されやすい。そこでポイント0以下のマスを避け、領域をできるだけ小さく分割するようにタイルを置いていく。この操作により獲得点数が高く、破壊されにくい領域を形成できるようにする。

3. 開発環境

OS:Windows,Linax

エディタ:Visual Studio,Unity,テキストエディタ

使用言語:C,C++,C#

18 INVADER

仙　台
（名取）
髙橋 諒大（4年）菊地　輝（4年）
日野 綾瀬（1年）北島 宏之（教員）

1. はじめに

今年の競技は昨年と類似しているが、エージェントの人数が増加し、フィールドが拡大されたことからより計算時間を要する。そこで、良手を高速に求める探索手法を用いて、サーバの状態と連携しながらも人間が操作する必要のないプログラムの開発を行った。

2. アルゴリズム

2.1 非公開フィールド

探索手法には、限られた時間内に比較的良い手が求められるビームサーチを採用した。幅広い盤面に対応できるように評価値を試行錯誤し、安定して高得点を狙う。

2.2 公開フィールド

非公開フィールドと同様にビームサーチを用いる。汎用的な評価関数では特色のあるフィールドに対して有利な手を安定して求めにくい。そこで、各フィールドに対して評価関数を特徴化させる。また、初動の定石研究を行い、最初数ターンにおいて最良の手を打てるようにする。

3. GUI と提出方法

GUI では、試合の状況が把握しやすいように点数表示とエージェント表示を分け、シンプルかつ視認性の高いことを心掛けた(図1)。サーバとの通信状態や探索状況もリアルタイムで表示し、プログラムの状況も把握できる。サーバとの通信は libcurl を用いて実装した。

図1 開発した GUI

4. 開発環境

C++17, VisualStudio2019, OpenSiv3D, libcurl

19 MOI

北九州
南　秀真（3年）岩本 圭介（3年）
大手 陸史（3年）松久保 潤（教員）

1. はじめに

今回、用いるプログラムはGUI部とアルゴリズム部、通信部および、これらを統括するシステム部からなる。

2. アルゴリズムについて

2.1 公開フィールドにおける事前の対策

モンテカルロ木探索に基づく考え方を利用し、勝利につながりやすいタイルを求める。

ランダムなプレイアウトを繰り返すことで、それぞれのタイルの重みを求める。

2.2 公開フィールドにおける手法

事前に求めたタイルの重みと 2.3 の非公開フィールドにおける手法で求めたタイルの重みの平均を、そのタイルの重みとして次の行動を決める。

2.3 非公開フィールドにおける手法

5-15 秒という限られた時間ですべてのエージェントの行動を決めるために、不要な探索の枝刈りが重要であると考えた。そこでビームサーチを用いる。

それぞれのノードにおける評価値はタイルポイントと領域ポイントの合計とする。

エージェントのいるタイルとエージェントに隣接する8つのタイルに対してビームサーチを行い、評価値が最大のタイルへ移動する。

3. その他

エージェントの行動をサーバーとやり取りするプログラムを組み込むことで、ソフトウェアの操作に要する時間や人的なミスを可能な限り減らす。

4. 開発環境について

IDE: Visual Studio 2019

OS: Windows 10

ライブラリ: OpenSiv3D

20 ＜ESC＞押したら異世界転生した件について

沖　縄
比嘉　風（3年）崎間 達帆（3年）
仲間　凛（2年）當間 栄作（教員）

1. はじめに

今回の問題では，試合開始時に情報が与えられる非公開フィールドを用いた試合と，フィールドの情報が事前に公開される公開フィールドを用いた試合が行われる．

2. 解法アルゴリズム

本チームが用いる解法は以下に示すとおりである．

非公開フィールドを用いる試合では試合の前半と後半で使用するアルゴリズムを切り替える．試合前半はビームサーチ法，後半はモンテカルロ法を採用した．

ビームサーチ法は枝刈りによって探索ノードを減らすことができる．そのため，探索空間が膨大な試合前半に適していると考え採用した．

モンテカルロ法は試合終了まで探索を行ってしまうためどうしても時間が掛かってしまう．そのため，探索ノードが減少している試合後半に適していると考え採用した．

公開フィールドでは非公開フィールドのアルゴリズムと同様のものを採用する．ただし，ビームサーチ法の評価関数はフィールドごとに用意し入れ替える．

3. 運営サーバーとの通信

ローカルサーバーを立て，そのサーバーを介して通信を行う．

4. 開発環境・動作環境

[OS] Ubuntu18.04 LTS, macOS Mojave

[Language] D, Scala, Go

[IDE/Editor] Vim, VisualStudio Code, IntelliJ IDEA

[Library] Scala FX

21 進捗log1

徳　山

吉武 拓海（2年） 広政 遼汰（2年）
齋藤 那月（3年） 力　 規晃（教員）

1. 概要

今回の競技は送られてきたフィールド情報をもとに2チームがマスを取り合い、ポイントを競う内容である。そのため私たちは自動で解を導いてくれる探索プログラムを作成することにした。

2. データの取得

JsonCpp を使いデータを読み込んでプログラム内で使用できるように変換する。

3. 探索方法

α－β探索、モンテカルロ探索のプログラムを用いて同時に探索し、よりよい探索結果を送信する。

4. GUI

図1に示すようなGUI操作画面を用いて、各チームのエージェントの位置、タイルポイント、エリアポイントを視覚的にわかりやすく、シンプルに表示する。

5. 使用環境

OS	Windows, MacOS, Ubuntu
使用言語	C++
開発環境	Visual Studio Code
ライブラリ	JsonCpp, Qt, Boost

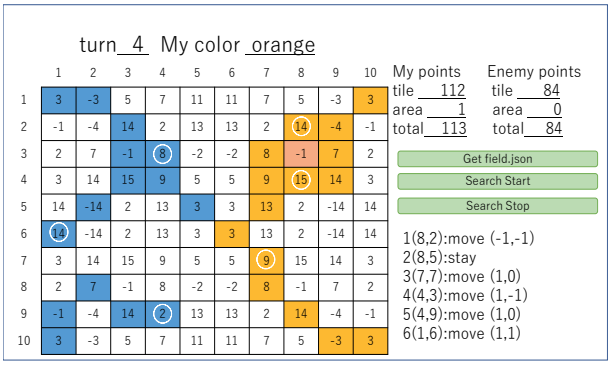


図1　GUI操作画面イメージ

22 高専の陣取り概論2

長　野

上松崇太郎（4年） 岩崎 凌汰（3年）
石田　 光（2年） 鈴木　 宏（教員）

1. 概要

本大会の競技は独自のゲームかつ、エージェントの数や盤面のサイズが試合によって異なるため、機械学習を用いたアルゴリズムを使用することにした。また、公開フィールドの対策や、計算時間を短縮する工夫も考案した。

2. アルゴリズム

機械学習を使用することによってこの競技の定石を見つけることや、様々な盤面に対応することが可能になる。機械学習にはディープラーニングと強化学習を利用する。それぞれ専用のライブラリを用いることで開発期間を短縮する。

3. 公開フィールドへの対策

今回の競技では予告された公開フィールドでの対戦が行われるが、それに特化したAIを作成したりはしない。代わりに、機械学習を進める段階で公開フィールドでの対局のデータを用いることで公開フィールドへの対策とする。

4. 試合の同時進行への対策

今回の競技は1ターンの時間が5～15秒と短い。さらに、最大3試合同時に進行する可能性があり、1回の手の計算にかけられる時間が少ない。そのため計算に時間がかかる木探索は学習時のみ利用し、本番のプログラムでは利用しない。

5. 開発環境

OS: Windosw10, macOS 10.14.3

言語: C/C++, Python

ライブラリ: OpenSiv3D, PyTorch, OpenAI Gym

23 天地を統べし海豹

秋　田

米田　凌（5年）鈴木 優也（5年）
藤原 滉太（3年）竹下 大樹（教員）

1. はじめに

効率の良いポイントの取り方を考えるのは、人力では困難である。そこで、盤面の情報や自分の得点などの情報をもとに評価関数や重み付け評価を使い、最適ルートの解析を行い、ゲームを進行させる。

2. 公開フィールドの戦略アルゴリズム

公開フィールドは、事前に盤面の情報がわかるため、あらかじめその盤面での試合をシミュレーション形式でコンピュータに行わせ、盤面の状況などの情報をもとに評価関数を作成し、最善手を導出するプログラムを作成する。

3. 非公開フィールドの戦略アルゴリズム

非公開フィールドは、その場ですぐに盤面の情報を把握し、最善手を計算させるという流れのため、最善手の導出が公開フィールドよりも時間がかかる。そこで、様々な状況に対応できるように得点の大きいマスを優先、領域ポイントを優先などのように複数の戦術を使い分ける。

4. 重み付け評価

最善手の導出の際には、重み付け評価を用いてエージェントの移動の優先順位を決める。盤面の得点やエージェントの場所、その周囲のマスの得点などの情報をもとに評価付けをし、最も重みの大きいマスを導出し、エージェントの行動を決定する。

5. 開発環境

開発言語 ： C, C++

使用 IDE ： Visual Studio

24 さんだぶ31

茨　城

里井瑠海奈（3年）鶴貝 拓海（3年）
緑川　惇（3年）安細　勉（教員）

1. システムの概要

JSON フォーマットのデータ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とそれをもとに HTML 回答フォームに自動で出力するプログラムをそれぞれ用いる。

まず、JSON フォーマットのデータを読み込み、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あたっては、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、HTML 回答フォームに入力。

2. 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とるまたは奪うことが可能なパターンを複数算出する。そのあとに、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。探索部のアルゴリズムにはモンテカルロ木探索を用いる予定である。また、接している面の数が減るほど（角や壁際に行くほど）、選択肢が減り、不利になると考えられる。そこで、それぞれに対応する重みをつけることによって、角や壁際に行きにくくする工夫もする予定である。また、評価値が一定以上低くなったら、そこから探索しないことにより時間の短縮もできるようにする。

3. 開発環境

Python3

25 矢野と愉快な仲間たち

長　岡
平田　蓮（3年）矢野 敦大（2年）
高橋　匠（1年）竹部 啓輔（教員）

1. はじめに

今年の競技は昨年のように人による対戦は完全に不可能である。また制限時間も 5 秒間と短いため Deep Q-Network を利用した強化学習を実装し、計算時間の短縮をすることとした。

2. 機械学習

盤面の情報を整理し、入力として使用する。3 に示すランダムマップジェネレータでランダムな盤面を作成し、学習モデル同士で対戦を重ね学習をする。

3. ランダムマップジェネレーター

機械学習を行うために、ランダムでマップを生成するプログラマを作成した。マップのサイズ、マスの配点、エージェントの初期位置をすべて乱数を使い決定する。これを使うことで、効率よく機械学習を行うことができる。

4. 開発環境

OS:Windows10/MacOS

言語:Python3.7.0/C++

IDE:Visual Studio 2019

ライブラリ:Kivy/ChainerRL

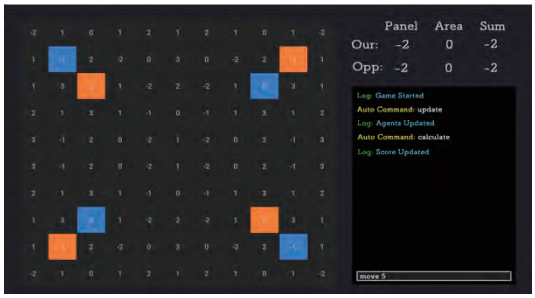


図 1　開発中のビジュアライザ

26 ＩＴの未来がここにあるっちゃが

大阪府大
吉田 凌河（4年）尾藤祐一郎（4年）
宮原 未来（3年）窪田 哲也（教員）

1. 概要

基本的にシステムは去年とほぼ同じで、公開フィールド用の評価関数と、アルゴリズムの改良を主に行った。

本番では、非公開フィールド用のアルゴリズムを使うが、評価関数を事前に作ったものにすることでより良い一手を選べるようにした。

2. ソルバーのアルゴリズム

2.1 非公開フィールド

アルファベータ探索に加えて、モンテカルロ木探索やビームサーチも同時に走らせることで、解の偏りを小さくした。

2.2 公開フィールド

各公開フィールドに対して、事前に公開フィールド用の探索アルゴリズムを用いて探索しておく。その結果を元に、ある盤面での評価関数を作る。また、盤面全体を通して明らかに悪いマスへの移動などを人力によって未然に防ぐ。

3. 特徴・工夫

複数のアルゴリズムを組み合わせてそれぞれの良さを活かしてる。本番では完全にソルバーに任せるのではなく、人間が次の一手を変更できるようにしている。

27 ドキドキエディター倶楽部

小　山

渡邊 海斗（3年）鈴木 朔矢（3年）
江口　晃（2年）干川 尚人（教員）

1. システム概要

今回の競技内容は Agent 移動により占領した領域の得点で競う。探索法には、最短経路探索、木探索などのアプローチが考えられ、我々はそれぞれの長所を状況に応じて使い分けることにした。

2. 探索法

2.1. 最短経路探索

最短経路探索はゲーム序盤に有効である。探索法には、今回の競技内容に合わせて改良した A*ベースのアルゴリズムを用いる。コストには移動回数、現在の得点、経路の重複度合などを用いる。ゴール候補を評価関数によりリストアップし、それぞれの候補に対して探索をかける。その中から最も評価値の高いルートを確定ルートとする。他の Agent は貪欲法などを用いて動かす。後の Agent になるほど、既に探索済みの Agent を確定ルートで動かすことにより探索の精度が向上する。

2.2. 木探索

木探索は混戦時に有効である。探索法には MonteCarlo 法または BeamSearch を用いる。探索時間、実装コストを考慮し採用を決める。また、木探索中でも探索対象外の Agent を確定ルートで動かす。

2.3. 探索法の選択

敵 Agent からの距離などの条件を用い、どちらの探索手法を使うか選択する。

3. 開発環境

OS　：LinuxMint、ArchLinux、Ubuntu
Editor　：Emacs、Spaceemacs、Neovim
Language　：C++、Python
Library　：GLUT、Matplotlib
使用機器　：ThinkPad、iiyama 8 シリーズ、Dell G3

28 class Snct：Yosano

仙　台
（広瀬）

伊藤 元斗（3年）赤垣 春樹（2年）
吉川慎太郎（2年）園田　潤（教員）

1. はじめに

今回の競技部門ではマスに配置された得点の量を考慮しつつ、より正確に多くの得点を取得するために、エージェントの数や 1 ターンの時間に合わせた適切な戦略アルゴリズムの選択が重要だと考えた。

2. 戦略アルゴリズム

戦略アルゴリズムとして主に用いる手法は、昨年度の競技部門において複数のチームで用いられた、「モンテカルロ木探索などで数手先の状態を見て、最適解だと評価された動きをする。」という手法である。

しかし、木の数が大幅に増加し、最適解が制限時間内に求まらない場合があるため、計算時間の管理が容易な改良型ビームサーチを用いたアルゴリズムや、貪欲法など、様々な戦略アルゴリズムを作成した。

3. 対戦シミュレーター

状況に応じて適切なアルゴリズムを選択するためには、様々なアルゴリズムを作成し、選択肢を増やす必要がある。対戦シミュレーターはエージェントのアルゴリズムの作成を支援するために作成された。

この対戦シミュレーターは、作成したアルゴリズム同士で対戦すること、作成したアルゴリズムと手動入力で対戦すること、手動入力同士で対戦することが可能である。

4. 開発環境

OS：Windows10
使用言語：C++, C#など
使用ライブラリ：Json.NET, DX ライブラリなど
開発ツール：Visual Studio 2019 Community など

29 おむおむず

大　分
田中 志貴（3年）小野 稜太（3年）
岩本 拓己（2年）徳尾 健司（教員）

1. 概要

エージェントの行動を決定するため、フィールドの状況からエージェントの行動を評価する評価関数を作成する。評価値をもとに数手先まで探索しエージェントの最終的な行動を決定する.

2. アルゴリズム

2.1 評価関数について

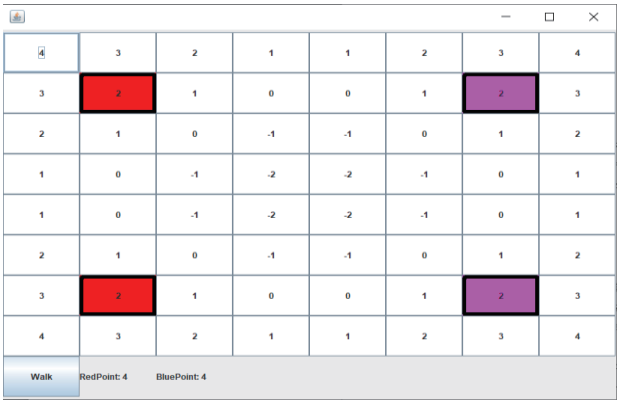
複数人で何度も対戦を繰り返し、対戦データを蓄積する.そのデータをもとにマスの得点、エージェントの位置、相手のパネル等のデータを引数にもつ評価関数を作成する.

2.2 探索部について

基本的には全探索を用いるが、時間短縮のため評価値の低い手は枝刈りして探索する局面を減らすようにする。

3. GUI

1 ターンの時間が短いので基本的には自動だが手動で操作する場合を考え、ボタン式にして視覚的にわかりやすく、操作しやすいようにした。



4. 開発環境

言語：Java　IDE：Eclipse

30 東京の高専IN品川

都　立
（品川）
若山 裕輝（3年）和田 靖広（5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、サーバとの通信を行う通信部分、エージェントの次の行動を決める処理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. 通信：サーバとの通信について

通信部分では、処理部分での結果をサーバで指定されている JSON 形式に変換しサーバに送信する。また、現在のゲーム状況を人間が理解するために、通信を行い取得したデータを可視化する。

3. 処理：アルゴリズムについて

ビームサーチを用いてエージェントの行動を決定する。ビームサーチは、行動後の盤面を評価し、評価の高い盤面でさらに行動させ評価することを繰り返すアルゴリズムである。今回は敵味方合わせて最大 16 体のエージェントが同時に行動する。すなわち、各ターンでの全てのエージェントの行動の組み合わせは 10^{15} を超えるため、一度に探索を行うことは難しい。よって、各エージェントの行動の枝刈りを行って盤面の数を削減する。

4 パラメータサーチ

処理部分で使用するビームサーチのパラメータを、機械学習を用いることで自動的に探索する。機械学習の手法としては、ベイズ推定を使用することによって勝率の一番高いパラメータを自動的に求める。

5 開発環境

言語: C++, Java, Python3

環境: Sublime Text 3, VSCode, Eclipse

31 君のマスもうないよ

阿　南

三河 多聞（4年）栗本 海音（4年）
武田 一磨（4年）平山　基（教員）

1. はじめに

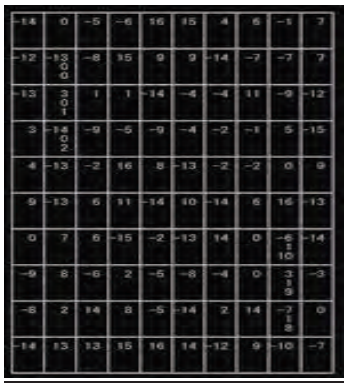
最適な行動を選択するため、探索の為に使える時間を十分に確保する必要がある。そのため選手による手動操作を行う場面を、極力なくすようシステムを構成した。

2. 探索アルゴリズム

エージェントの行動を決定するアルゴリズムを複数用意し、状況によって適切なアルゴリズムを選択できるようにした。同じフィールドにいるエージェント同士で違うアルゴリズムを採用することも可能である。また、エージェントのアルゴリズムとは独立して、最終ターンまでの行動を探索するアルゴリズムをエージェントのアルゴリズムと並行して動作させる。このアルゴリズムが探索し終えた時点で、エージェントの制御が移行される。

3. 公開フィールド対策

最終ターンまで探索するアルゴリズムによって、試合開始までに可能な限り探索を行い、結果をデータ化することで有利な状況で試合を開始することができる。



4. 開発環境

言語：Java、C++

エディタ：Vim、Visual Studio Code

32 秒速50km／sで移動する点P

津　山

瀧本 隼矢（3年）友末 智将（2年）
下山 朗弘（2年）松島由紀子（教員）

アルゴリズムについて

1 エージェントの行動確定アルゴリズム

・自チームの移動可能マス各評価点を列挙し、それぞれのマスについて評価関数を用いて評価点をつける。評価関数の評価点には優先度による重みをつけ、最終的に評価点が最も高い手を選択する。
以下に評価の優先度の高い順に示す。
1. 領域ポイントが取れそうかどうか
2. 相手の領域・タイルポイントが高いかどうか
3. 移動した後の次の移動可能マスが多いかどうか
4. 移動先のマスの点数が高いかどうか

・2. の評価優先度は、相手領域ポイントが自分のポイントより高いときに相手タイルと近い自エージェントのみ高くし、遠いと低くする。

・エージェントの先読みターンは序盤から中盤までは基本的に 4 ターンとする。先読みターン数はエージェントの数により変動する

1.2 公開フィールドでの戦略アルゴリズム

事前対策で得た良い手を求めるアルゴリズムを使用

1.3 日公開フィールドでの戦略アルゴリズム

マイナスが多い範囲は領域ポイントを、プラスが多い範囲はタイルポイントを取ることを中心に動く

2. 開発環境

【OS】Windows10, 7

【Language】C, C++

【Frame Work】Siv

【IDE】VisualStudioCommunity2017

33 独立行政法人国立高等専門学校機構東京工業高等専門学校

東　京

柴田 紘希（3年）岡田　航（3年）
山口 翔太（2年）山下 晃弘（教員）

1. GUIについて

このプログラムのUIでは中心にフィールド、右上にターン数、右下と左下にそれぞれのチームの現在のタイルポイントと領域ポイントをそれぞれ表示している。現在のポイントの文字の色とチームの色を一致させることでどちらのチームがどれ程のポイントか一目でわかるようにした(図1)。

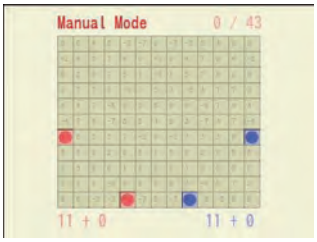


図1 プログラムのスクリーンショット

2. 書式について

このプログラムではオートモードと手動モードを実装した。手動モードでは全員のエージェントの行動を手動で選択してゲームを進めることができ、オートモードではアルゴリズムによってゲームを自動で進めることができる。手動モードはデバッグに使用し、本番ではオートモードを使用する。これによって開発段階には人間側でしっかりとデバッグを行うことを可能にし、本番ではアルゴリズムにより、良い手を見つけられるようにした。モードはキー一つで切り替えることができるようにし、図1の左上の”Manual Mode”のように表示させることで、今は何のモードなのか人間側が見て一目でわかるよう工夫をした。

3. アルゴリズム

アルゴリズムにはBeamSearchやMin-Max探索を組み合わせたものを用いて次のターンの行動を探索した。

4. 開発環境

通信関係ではC++とPythonを組み合わせて実装し、UIはC++とQtを用いて、競技者が実際に使用しながら改善した。

34 蜘蛛忍者

佐世保

秋永　晶（3年）坂本 光太（3年）
髙田 康介（3年）嶋田 英樹（教員）

1. はじめに

今回の競技部門は、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。加えて、公開フィールドと非公開フィールドの二種類があるため、それぞれに効率的な探索方法を適用する。

2. アルゴリズムについて

2.1 非公開フィールド

モンテカルロ木探索により、ランダムにエージェントを試合終了まで行動させ、ゲームに勝利する可能性の高い手を検討する。

2.2 公開フィールド

深層学習を用いた評価関数を作成し、ゲームに勝利する可能性の高い手を検討する。学習には公開フィールドでの条件下での自己対戦結果を用いる。

図：開発中のシステム画面

3. 開発環境

［言語］Java・Python

［IDE］Eclipse・JupyterNotebook

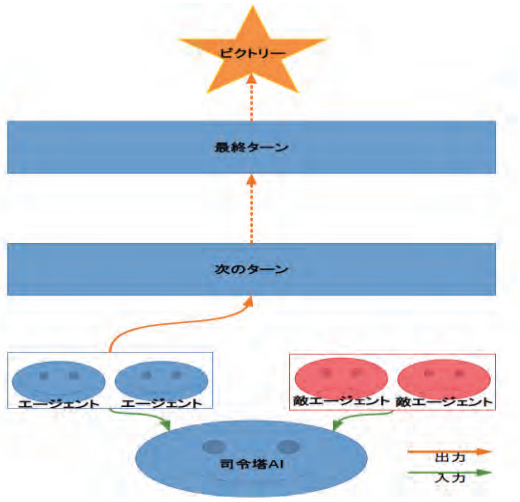
Gogle Colaboratory

35 T. F. Revolution

和歌山
河邉 渓介（3年） 明石 堅斗（2年）
松岡 裕希（2年） 森　　徹（教員）

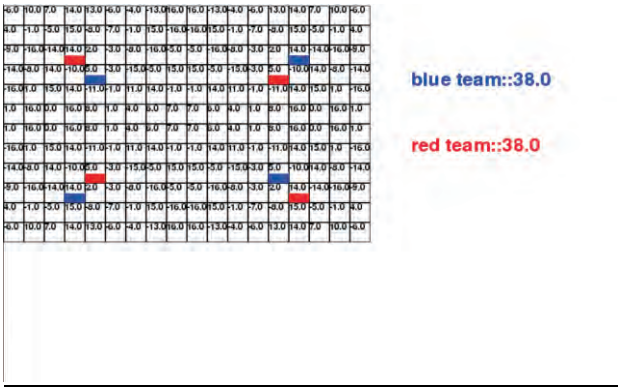
1. アルゴリズム

毎ターンごとに、相手の動きと自分の動きを入力して、最終ターンまでに相手より多くの点を取れるように、自エージェントの動きを出力する。



2. UI

無駄をなくし、簡単に操作するためにシンプルなUIになるよう開発にした。



3. 開発環境

言語:Python(pygame, Tensorflow)

開発環境:Visual studio2019

Visual Studio Code

36 tres

都　城
青木 琢磨（2年） 倉永 隆太（2年）
濵川　黎（2年） 御園 勝秀（教員）

1. 概要

両チームの手をランダムに決め、その繰り返しによって手を探索するモンテカルロ木探索と、対戦データをもとにした強化学習によって作成した評価関数を軸に指示決定を行う。

2. アルゴリズム

2.1 モンテカルロ木探索

モンテカルロ木探索のおおまかな動作は以下の繰り返しが基本である

1．ランダムに手を選択し、木を降りる

2．末端でノードを作成する

3．一度プレイアウトを行う

4．結果を更新する

2.2 手の選択について

ランダムに手を選択するモンテカルロ木探索だけではその手が本当に有力な手かどうか完全には分からない。そこで UCT というものを導入する。これは、プレイアウトの結果とその手の期待値がどのくらいかをもとに行っての価値をきめるものである。

2.3 公開フィールドと非公開フィールドでの行動

今回は公開フィールドと非公開フィールドがあるため、それぞれについての行動の仕方を考えた。公開フィールドでは、そのフィールドを使った対戦データをもとに強化学習を行い、できるだけ効率の良い囲み方を算出した。非公開フィールドでは、効率の良さそうな囲み方を自動で解析するようにした。

3. 開発環境

言語　C++、エディタ　Vim

37 これを書いている明後日は面接なのですが大丈夫なんでしょうか

宇　部
青木 勇大（1年）堀江 裕貴（5年）
村上 航輝（2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，手番中に取る行動の決定に用いるアルゴリズムの概要を示す．

(1) サーバーより受信できるJSONデータより，フィールド上のマスのデータを操作するPC内のデータを現在のフィールド状況に更新する．

(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しサーバーに情報を送信する．

(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(1),(2),(3)を繰り返し更新する．

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前回のプロコン競技部門のプログラムからさらに効率化が必要になる．

3. 開発環境

Windows10, ArchLinux

38 null

沼　津
村尾　響（3年）相澤 大和（2年）
西秋 大輔（2年）鈴木 康人（教員）

1. はじめに

今回の競技内容は前回の競技と内容が変わらないため、去年のソースコードを見直す形で開発を進めた。去年とのルールの差分である競技サーバーとの通信システムを作成し、全体的にオブジェクト指向のもとクラスを作成し再コーディングを行った。

2. 非公開フィールドへの対策

非公開フィールドでは、数ターン後の手を全探索して、その中で最もポイントを取ることができる手を探すことで常に最善手を取ろうとするミニマックス法を用いる。

3. 公開フィールドへの対策

公開フィールドでは、あらかじめコンピュータ同士で対戦させ、マスごとの評価値を得てから実戦を行う。その評価値の取り方はランダムにエージェントを動かし、その対戦での勝敗やマスごとのタイルの有無によって評価値を調整する。この対戦を複数回行うことで評価値を決定し、実戦でのエージェントの移動の参考にする。

4. 開発環境

言語 ： C++, C#, Python3

環境 ： Visual Studio, Visual Studio Code, Vim, GCC

ライブラリ ： TensorFlow, Chainer

39 あんあ～ていふぃしゃる

鈴　鹿

立松 諒也（5年） 濵口 翔吾（3年）
豊田 真吾（3年） 田添 丈博（教員）

1. はじめに

今回の競技は昨年と同様対人戦となっており、相手の行動や現在のフィールド状況などにより、その時の最適行動を一定の理論に基づいて選択することは困難なため、機械学習を用いて最適行動を選択するプログラムを作成する。昨年と違い今年はエージェントの数が 2 体より多くなる場面があるため、全てのエージェントの行動を出力するモデルを作成すると、そのモデルの学習が著しく遅くなる可能性がある。そこで、例えば味方のエージェントが 4 体なら、1 体の行動のみを決めるモデルを 4 つ作成するといった工夫を行う。

2. モデルについて

2.1 概要

1 つのモデルに多くのエージェントを操作すると学習が上手く進まない場面があったため、エージェントの数に応じてモデルを複数用意することにした。以下の図 1 がそのイメージ図である。

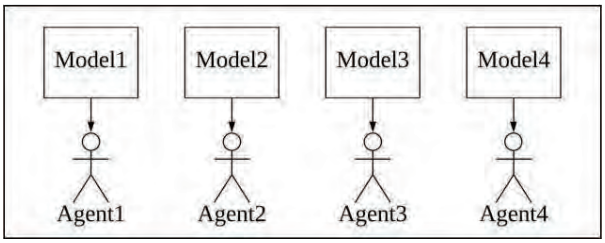


図 1 イメージ図

2.2 モデルの出力について

機械学習では基本的に評価が高い行動を取るようにするが、その行動が範囲外を行動するなど無駄な行動だった場合、その次に評価が高い行動を取るように実装する。

3. 最後に

データの取得、行動の決定、回答は全てプログラムが行うように実装する。

40 ねこさん同好会

岐　阜

長尾 颯真（3年） 苫米地康太（3年）
杉浦 颯斗（3年） 廣瀬 康之（教員）

1. はじめに

ゲーム終了までの盤面を全探索するのが最善であるが、1 ターンで遷移可能な状態が多すぎるため全探索は不可能である。そのため探索範囲を狭めて、相手チームの駒の動きを考慮することなく、現盤面から自チームの駒を動かすことにより獲得しうる最大得点を得るよう、効率よくゲームを運ぶようにすることを意図する。

2. 戦略

自チームの持ち駒が多ければ相互に連携し、市松模様を作るなどして領域点を多く獲得する。また駒が少なければ高得点マスを効率よく塗りつぶすという戦略を用いる。どちらにしても駒同士での連携、無駄な動きを作らないという事に主眼を置いた戦略を採用した。

3. アルゴリズム

今回のゲームでは 1 ターンの時間が試合によって変化するためビームサーチの亜種である chokudai サーチと呼ばれるものを使用する。これはビームサーチと比べ時間管理がしやすく、探索結果で似通ったものがそこそこに抑えられ多様性がある程度確保できるため、無駄な探索を省くことが出来ると考えられる。

評価関数については、単純な得点だけでなく、残りターン数とそのターンで到達できる領域を加味した予測値を準備し用いることで、最終に自チームの高得点につながる手順を導き出すことが出来ると考えている。

4. 開発

C, C++

41 国立高等専門学校機構釧路工業高等専門学校創造工学科スマートメ

釧　路
泉　知成（2年）村上　渚（2年）
泉谷 春樹（2年）天元　宏（教員）

1. はじめに

今回は公開フィールドと非公開フィールドで異なる探索、アルゴリズムを使用し効率よく行動ができるように構成をした。

2. アルゴリズム

2.1 A＊アルゴリズム

公開、非公開フィールド内の最高得点を最速で獲得することにより自チームの得点となるだけでなく、敵チームのエージェントが獲得を試みる場合、妨害することができる。

2.2 遺伝的アルゴリズム

移動する方向を遺伝子とした個体を乱数で複数生成し、動いたマスの合計点をその個体の評価値とする。局所解にならないように突然変異をさせるが、どれくらいの確率で突然変異を起こすかがカギとなると思われる。適当な世代まで繰り返して最も獲得点数の高かったものを扱う。

2.3 モンテカルロ木探索

ランダムに回数を重ね探索し、非公開フィールドでも正常に行動ができるようにプログラムを構成した。

相手の存在を考慮しない自分のみの理想的な状況で学習させる。相手の行動に対処するプログラムは別途用意する。

3 開発環境

Python3, Sublimetext, Windows10, OSX10.14

42 踊らないエージェントは左遷

八　戸
小倉 直弥（4年）伊藤 瑠威（4年）
菊地 大樹（3年）細川　靖（教員）

1. システム概要

事前に用意したアルゴリズムを 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させ、各エージェントの進行方向を決定していく。公開フィールドでは改良を加えたアルゴリズムを用いる。

2. ベースアルゴリズム

Minmax 法をベースとしたアルゴリズムで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。

3. 公開フィールド・非公開フィールドでの調整

公開フィールドでは各フィールドに対し事前に分析を行う。得られたタイルの点数の偏りを元に評価関数を調整しアルゴリズムの補完を行う。非公開フィールドではフィールドを解析した瞬間に評価関数を調整する。

4. GUI

盤面をわかりやすく視覚化するために GUI を用いる。フィールドの情報の他にフィールド内におけるタイルの点数の偏りなども表示する。また、複数試合同時進行にも対応する。

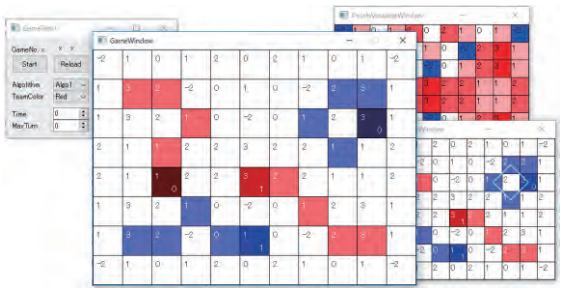


図　GUI の例

5. 開発環境

言語：C/C++　ライブラリ：Qt

IDE：Qt Creator Open 4.6.0 Community

43 Procon30移動距離部門 決勝戦敗退

旭　川

齋藤 唯翔（4年）近江 雄哉（5年）
加納 源基（2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 によって現在の状況を表示し、操作者が確認できるようにする。

2. アルゴリズム

外側からより多くの領域ポイントを取ることを軸としたアルゴリズムを作成する。相手領域が周囲に 3 つ以上ある場合は、タイル除去を優先する。

3. UI

図 1 のように、サーバから得られたマップの情報を GUI 上にグリッドで表示し、毎ターンの開始時に通信を行って現在の位置情報や取得タイルを受け取り、更新を行う。

図 1 GUI の例

4. 開発環境

OS	Windows 10
IDE	Visual Studio 2019
言語	C++、C#
ライブラリ	Json.NET、.NET Framework 4.7.2

44 セグメンテーションフォルトベクターサブスクリプションアウトオ

豊　田

鶴田 拓磨（5 年）長坂 光将（4 年）
三井 敦稀（3 年）藤原 賢二（教員）

1. システムの概要

今回の競技では、サーバから取得したフィールド情報をもとに、エージェントの行動を決定し、行動情報をサーバに送信する。使用するアルゴリズムは公開フィールド用と非公開フィールド用の 2 種類を作成する。サーバとのやり取りは Python3 を使用して行う。

2. 学習方法

深層学習を用いて強化学習を行う。強化学習では、両チームの得失点を報酬とし、行動の結果を自分自身で判断することでよりよい行動をするよう学習を行う。様々な対戦データに対して深層学習を行い、モデルを構築する。構築したモデル同士での対戦のみではなく、人によって作成したアルゴリズムとの対戦も行い複数の戦略を学習する。

各公開フィールドごとに学習を行い、各フィールドに対して最適なモデルを構築する。非公開フィールド用のモデルは公開フィールドから構築されたモデルを利用して、ランダムに生成した複数のフィールドで対戦を行い学習する。

学習を繰り返す中で AI の学習が適切に行われているか、過学習が起こっていないかを確認するために、フィールドの状況を GUI で表示する。

3. フィールドの表示

学習状況を確認するために Python3 の Tkinter を使い、図のように競技フィールドを表示する。

12	3	5	3	1	1	3	5	3	12
3	5	7	5	3	3	5	7	5	3
5	7	10	7	5	5	7	10	7	5
3	5	7	5	3	3	5	7	5	3
1	3	5	3	12	12	3	5	3	1
1	3	5	3	12	12	3	5	3	1
3	5	7	5	3	3	5	7	5	3
5	7	10	7	5	5	7	10	7	5
3	5	7	5	3	3	5	7	5	3
12	3	5	3	1	1	3	5	3	12

図　公開フィールド A-1 のフィールド表示

4. 開発環境

C++（g++14），Python3, ChainerRL

45 main=putStrLn "Hello World"

石　川

吉岡 龍吾（4年） 喜井 遥介（4年）
谷内　亘（4年） 松本 剛史（教員）

1. システムの概要

Haskell で書かれたメインプログラムが回答システムとの JSON の送受信を行い，C++で書かれた次ターンのエージェント動きを求めるプログラムを呼び出す．このプログラムは領域を分割することで，列挙する状態数を減らし，数手先までの読むことを可能とする．スコア計算のプログラムでは，タイルポイント，領域ポイントの計算とあと数手で領域ポイントが得られるタイルを求め，それを次手の探索で利用する．図 1 にシステムの概要図を示す．

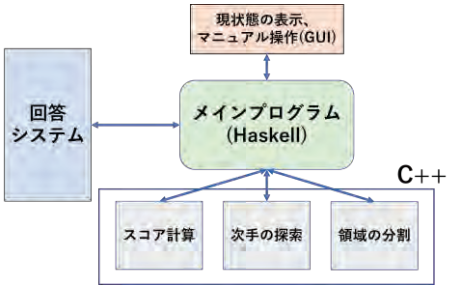


図 1　システムの概要図

2. GUI について

GTK を用いて盤面の状態，自チームと相手チームのエージェントの位置とスコアを表示させるプログラムを作成した．プログラムが完全でないことを考え，人間が GUI を通して，次ターンの指示を入力する機能も実装した．図 2 に GUI 画面の一例を示す．

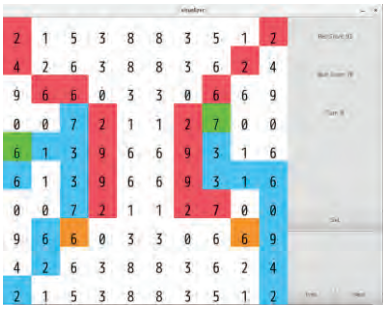


図 2　作成した GUI 画面の一例

46 com−PASS

弓削商船

小井川秀斗（2年） 金山 裕星（2年）
笠松 辰治（1年） 長尾 和彦（教員）

1. はじめに

今回の競技部門は、フィールド上にあるタイルポイントを取得しながら進み、タイルを囲んで領内のポイントを取得して相手とポイントの合計値を競う陣取りゲームである。

2. 戦略アルゴリズム

ゲーム中に盤面の探索を行うことで行動を選択する。探索にはビームサーチに基づくアルゴリズムを用いる。ビームサーチで用いる枝刈りには、スコア上位５場面とランダムの選ばれる２場面のみを残すという方法を用いる。これは幾らかのランダム性を持たせ、行動の偏りが生じる事を防ぐ目的がある。

3. コンピュータの振り分け

昨年度と違い同時に複数の試合を行う可能性があるため、３台のコンピュータを用いた並列処理を採用する。探索プロセスを管理する親と、実際に探索を行う子を用意し、重要度に応じてリソースや実行時間を割り当てる。負けている試合ほど重要度を高くすることで探索量を増やし、逆転を狙う。

4. 開発環境

プログラミング言語：Python3, Rust, C++

主な使用ライブラリ：Tensorflow, Pytorch, Numpy, Qt

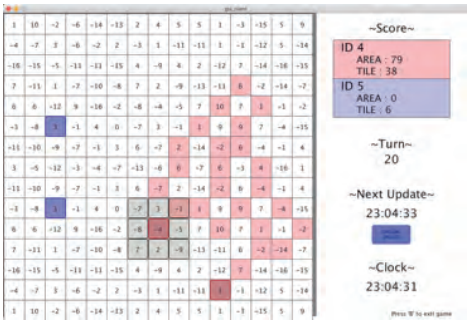


図１シミュレータ

47 重版!!Meglimath 第三版

富　山（射水）

小橋 音徳（2年）大道 太陽（2年）
織田 祐斗（2年）門村 英城（教員）

1. 概要

本プログラムはサーバからフィールド情報を取得した後、公開フィールド及び非公開フィールドそれぞれに適した戦略アルゴリズムを用いて多くの得点を得ることを目的とする。

2. 公開フィールド

エージェントからの距離が離れているフィールドのマスの情報にプーリング処理を行うことでマスの情報を簡略化する。情報の簡略化が行われたフィールドの得点を、ベルマンフォード法のノードに向かう経路の重みとし、多くの得点が集まっているマスに向かうための最善手の方向を求める。

2.1 公開フィールドへの事前対策

フィールドのプーリング処理に使用する計算方法とプーリングの範囲が異なるものをあらかじめ用意しそのパラメータを用いてコンピュータ同士で対戦させた結果から本選に使用するものを複数選ぶ。

3. 非公開フィールド

高い得点を得るためには領域ポイント、タイルポイントを多く獲得する必要がある。特に領域ポイントを効率よく獲得するために、エージェントをジグザグと段々に動かす。壁の有無や得点からエージェントの行動を計算する。

4. GUI

試合を進める上で進み具合や試合情報、時間等をリアルタイムに把握できるのが良いと考えられる。そこでそれらの情報を表示し、尚且つ画面上で操作できるようにする。

5. 開発環境

言語：Python 3

エディタ：Visual Studio Code

OS：Windows 10

48 どうせ去年と同じモンテカルロ木

奈　良

比嘉 隆貴（3年）藤本　祥（5年）
徳舛 迪也（2年）山口 賢一（教員）

1. 概要

本プログラムでは，あらかじめ機械学習を用いたモデルによって，エージェントが操作される．送受信部とゲーム盤面とエージェント指示部により構成されている．

2. 指示決定アルゴリズム

公開フィールドでは主に強化学習を用いたモデルにより指示決定を行う．未公開フィールドでは，ランダムマップと機械同士の戦いを混ぜた学習データで構築されたモデルを使用する．しかし，人間と対戦する場合と機械と対戦する場合で評価が異なってくる場合があるので，別でモンテカルロ木による探索も行い，良い評価の選択を行うこととした．（パンフレット作成時点のアルゴリズムであり，試行錯誤しながら改良を行っているので本番ではアルゴリズムが異なる場合があります）

3. シミュレーター

学習時に使用したシミュレーターと，本番で使用するシミューレーターのコア部分は同じである．

4. 開発環境

OS: Arch Linux
Language: C/C++, Python
Editor: Neovim, VisualStudioCode

49 いちげきひっさつ ゆびをふる

舞　鶴
久万颯一郎（5年） 河原 未侑（4年）
中島 滉太（3年） 舩木 英岳（教員）

1.　はじめに

今年度の競技では、昨年度の競技と同様に、限られた時間の中で点数配置や進行状況を考慮した、最適な行動を算出する必要がある。よって、我々は AI を用いて最も評価の高い解を算出し、その解をターンの行動として送信する。

2.　システム概要

2.1　データの入力

作戦ステップの最初に、回答システムからテキストファイルでデータを取得する。データ取得も制限時間に含まれるので、後述の司令塔の指示決定アルゴリズムに素早くデータを伝える必要がある。

2.2　司令塔の支持決定アルゴリズム

4 手先までの行動で得られる可能性のあるタイルポイントを計算し、獲得できるタイルポイントが一番多くなる行動パターンを選択するアルゴリズムを基本として、深層強化学習を用いて公開フィールドを含めた様々なフィールドに対応できる AI を作成し、過去に学習した状況から最適な行動を算出する。

2.3　データの出力

算出した行動を JSON フォーマットのデータに変換して回答システムに送信する。正しくデータ送信が行われなかった場合、そのターンの行動が無効となるので、書式などに間違いがないか、確認するシステムを設ける。

3.　開発環境

OS:Windows10 ， バージョン:1803

Microsoft Visual Studio 2019, Dx ライブラリ,

Unity 2019, Anaxonda3, ml-agents 6.0

開発言語: C++, Python 3.6

50 ごみ拾いは楽しいね

木更津
田中　幸（3年） 高名 典雅（3年）
八木 遼介（3年）
サプコタ アチュタ（教員）

1. 公開および非公開フィールドにおける戦法

この競技において相手に多く差を付けられるタイルの取られ方の一つとして、味方の色で塗られたタイルが敵エージェントに除去されて高得点タイルを敵の色に塗られてしまうことである。

そのことから、高得点タイルをただ塗っていくより、高得点タイルが密集している領域を囲んでさらにその周りを塗って敵のエージェントからタイルを奪われにくくする戦法が良いと考えた。

また上述した戦法では味方エージェント全員が同じ目的を持って動くが、もう一つの戦法として各エージェントに役割分担させるジョブ戦法が存在する。分担の種類としてはまだ塗られていない高得点タイルを確保する担当や相手の高得点タイルを奪いにいくという担当があり、作業を分担させることで相手に点数差をつけやすくすることができる。

これら２つの戦略を駆使して戦っていく。

2. シミュレーション画面

図１のようなシミュレーション画面を作っている。

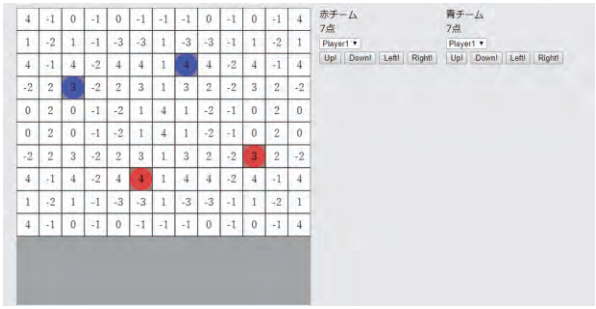


図１　シミュレーション画面

51 シャローラーニング

鶴　岡
五十嵐拓真（3年）佐藤 勇人（3年）
伊東 和磨（2年）金　帝演（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。この競技を攻略するには、タイルポイント、領域ポイントをどのように取るかが重要になってくると予想される。そのため、それぞれのポイントを取るタイミングと取り方について考えプログラムを作成する。

2. フィールド分析

フィールドの点数分布から高得点が密集している区域を求める。このとき、3×3 の区域ごとにタイルポイントの合計を求めて最も高い区域を高得点区域とし、この区域にエージェントを誘導することで効率的にタイルポイントを回収することができる。

3. メインアルゴリズム

エージェントの動きを決定するアルゴリズムは「タイルポイント優先」「領域ポイント優先」「相手との合計ポイントの差」の3つがある。

序盤は「タイルポイント優先」で高得点のタイルポイントを回収し、中盤から「領域ポイント優先」で領域を作り、終盤は「相手との合計ポイントの差」を比較することで、点差が広がりやすい領域の破壊や領域の生成につなげる。各アルゴリズムの切り替えは総ターン数に対する割合によって決定する。

3つのアルゴリズムはいずれも各エージェントの 3 手先まで最も高得点になる経路を求め、エージェントを移動させる。

4. 開発環境

開発環境　：Visual Studio 2017
OS　　：Windows 10
言語　　：C#

52 パワーオブチェリーアイランド

鹿児島
野間 隆眞（2年）引地　涼（2年）
窪田将太朗（2年）原　　崇（教員）

1. エージェントの行動決定

アルゴリズムによって周囲 8 マスの重みを計算し、一番重みが重いところに進む。

アルゴリズムはエージェントからの距離、マスのポイント、そのマスに進んだ時の周囲 8 マスのポイント、敵タイルとの位置関係などの多くの要素の関係から計算するものである。

2. GUI

マップの状態を把握しやすいように GUI（HTML、CSS、JavaScript で構成されている）上に出力した。2 チームの点数やエージェントの行動を表す json はブラウザのコンソールに出力するようにした。

盤面の状態の例を図 1 に示す。

また、フォームに入力された json 文字列からマップを生成する機能も搭載している。

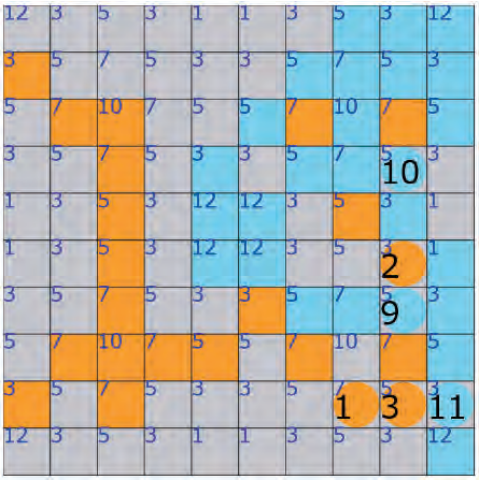


図 1　GUI

3. 開発環境

言語：HTML, CSS, JavaScript
エディタ：VS code, Atom
ブラウザ：google chrome, Firefox

53 Unlimited Debug Works

高　知
岡本 和樹（4年）長澤　陸（4年）
公文 裕太（4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プログラムで行動を決める。

2. 公開フィールドの行動決定

行動を決定するアルゴリズムを持つプログラムを複数作成し、得点を取ることをプログラムにとっての評価とすることで、高い評価を持つプログラムがより良い行動するプログラムだと考えられるので、今回は高い評価を持つプログラムを採用する

3. 非公開フィールドの行動決定

機械学習を基にした AI を用いる。複数の行動を決定するアルゴリズムと機械学習をしたものとを競わせ、機械学習の精度を高めていくことで、どんな非公開フィールドでも高評価の行動決定を導くことができると考えられる。

4. データの入出力

マスのタイルポイント、領域ポイントとエージェントの位置をサーバから送られる情報 json-c で読み取り、プログラムに出力する。プログラムにより行動決定した情報を簡易回答システムでサーバに送信する。

開発環境

OS: Windows 10

IDE: Visual studio code/Visual studio 2019/anaconda

言語:C/python

54 SonarPlant

広島商船
平尾　光（3年）藤野 元規（3年）
和泉 和実（2年）岩切 裕哉（教員）

1. はじめに

本システムは、試合状況に応じた複数の作戦を用意しておき、各エージェントに最適な作戦をその都度与えて実行していくものである。

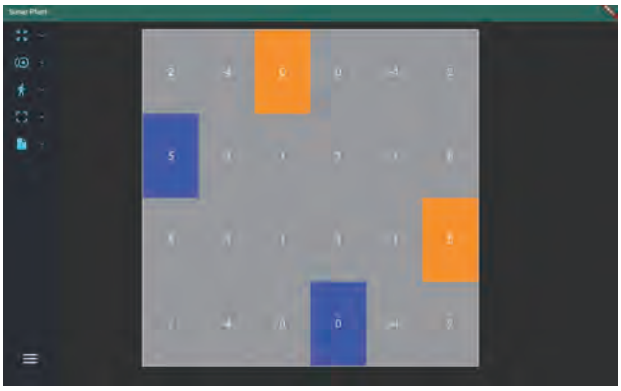
2. 今回の作戦の考え方について

今回は、作戦として、柔軟に、また正確・確実に領域を獲得できるように、複数の作戦を後述の「性格」という形でプログラムに自動で選択させ、対戦に使用する。

また、プログラムで場の状況を判断させて大まかな戦術を選択させることで、迅速性・確実性を増すことを考えた。

2.1 性格

性格については、前述したように、複数の作戦を「性格」として見立てて自立させること、場の状況に応じて大まかな戦術を変えるといった構成にすることで、柔軟性があり、正確で確実に領域を獲得することが可能だと思っている。



（図）対戦用ソフトウェアのイメージ

また、公開フィールドでは事前に対策をすることでプログラムの取れる選択肢増やすことを優先し、それによって「性格」にも影響を与えることで合理的な動きが出来ると考えた。

3. 開発環境

Flutter, Dart

55 場ザップ!～すべての陣地を獲得する裏技～

有　明

福田 一心（3年） 西島 佐介（4年）
大山 真弥（4年） 森山 英明（教員）

1. 概要

本システムでは、フィールドデータから各チームの取得しているマス、およびエージェントの位置を分析し、自チームのエージェントが移動すべきマスを計算する。データの取得、計算、および回答の送信を自動化することで、プレイヤーの介入を必要としないシステムの実現を目指す。

2. 戦略

以下の 2 つのアルゴリズムによる計算を行う。

(1)非公開フィールドにおける動作：

自チームの各エージェントにおいて、現地点から 4 手先までに獲得可能な得点を計算し、最も高い得点を獲得できるマスを解として送信する。相手エージェントの囲い込みの妨害、および自エージェントの囲い込みに繋がる移動を高得点として計算する。なお、囲い込みの判定は、エージェントの周囲 3 マスにおいて、自チームのマスが隣接している相手チームのマスの数が一定数を超えた場合に、囲い込みとみなす。

(2)公開フィールドにおける動作：

非公開フィールドでのアルゴリズムに加え、各マスの点数を考慮して得点を計算する。例えば減点マスが多いフィールドでは、囲い込みによる点数獲得が重要になると考える。

3. プレイヤーへのサポート機能

システムの動作の不具合が発生することも考え、プレイヤーに進行状況を分かり易く表示する UI、相手チームによる囲い込みの警告機能を実装し、プレイヤーの手動操作をサポートする機能も実現する。

4. 開発環境

使用言語:java、pyhton

IDE:processing、Visual Studio

ライブラリ:Jackson、ControlIP5、kivy1.11.0

開発環境:Windows10(64bit)

56 TMCIT　Arakawa

都　立
(荒川)

田中 隼人（4年） 佐藤 真夕（4年）
米山 胡桃（4年） 齋藤 敏治（教員）

1．はじめに

今年の競技部門の種目はタイルに区切られたフィールド内でいかに多くのタイルを占領し、また限られたターン数でいかに多くの点数を獲得できるかという陣取りゲームである。さらに、このゲームでは一回のゲームでエージェントが何体配置されるかもわからないため小数、多数それぞれの場合でも対応できる効率的なルートの探索方法を実装した。

2．概要

我々はゲーム攻略のための探索方法として「ツリー構造」を軸に考えられる複数ものルートの中で最短かつ高得点を獲得できるものを見つけ出すアルゴリズムを考案した。相手の動きに対し臨機応変に対応できるよう、試合中に何度もさまざまな探索経路を導き出しその度試合状況に応じて最も適したルートを選択し進んでいく。

こうすることで、敵との接触回数が抑えられ効率的かつ柔軟に試合を進め得点を稼げると考えた。

3．開発環境

[OS]windows7/8/10

[言語]Python/java/C

57 GO GO GO!!!

苫小牧
佐竹 航希（2年） 伊藤 成理（3年）
吉川 千晴（2年） 土居 茂雄（教員）

1. はじめに

今回の競技はターン制のため，ゲームルールの処理を行うサーバを用意する．また，探索，回答システムとの通信もサーバで行う．そのほか，盤面状態の表示，およびサーバの操作を行う UI を用意する．

2 各フィールドの戦略

2.1 公開フィールド

あらかじめ公開フィールドのゲーム木を，α-β 法を用いて求め，それぞれの盤面に対して評価を行う．その結果を用いて次のステップの行動を決定する．

2.2 非公開フィールド

フィールド全体のそれぞれのマスを優先度順に評価し，数手先までを探索して最も結果の良いルートを求める．

マスの評価は，正のマスの値の平均値を基準となるしきい値とし，しきい値以上で敵が取得しているマス，しきい値以下で敵が取得しているマス，負のマスなどと分類する．これを用いて優先度順に評価を行う．

3. システム構成

3.1 サーバ

サーバは，回答システムとの通信，盤面情報の保持・処理，探索を行うソルバー，UI との通信を行う API を実装する．盤面情報の保持には RDB を用いる．また，公式の回答システムを用いずに試合を行うスタンドアロンモードを実装する．

3.2 UI

UI は，サーバとの通信を行う API インターフェイス，接続情報等の保持，盤面情報の表示部分を実装する．

4. 開発環境

言語：Go

エディタ：Visual Studio Code