進化計算とUCTによるMarioを人間らしくプレイするAI

中野 雄基[1,a)]　美添 一樹[2]　脇田 建[3]

概要：人間らしいプレイをするAIは，テストプレイヤーの労力の軽減，人間と協調してプレイするAIの作成などに有用である．本研究では遺伝的アルゴリズムとモンテカルロ木探索を組み合わせ，人間に似たプレイをするAIを自動的に作成することを目的として研究を行った．
Mario AIにおいて人間の操作履歴を元に学習し，模倣するAIを作成し，どの程度うまく模倣できたか，人間のプレイ履歴とAIの行動との一致率によって比較した結果，最大で52%の一致率を達成した．また，数人の被験者に対して，人間のプレイと区別可能かどうかアンケートを行った．

Humanlike AI for Mario Bros. Based on Evolutionary Computation and UCT

YUKI NAKANO[1,a)]　KAZUKI YOSHIZOE[2]　KEN WAKITA[3]

Abstract: Human like AI for games would be useful for alleviating efforts of test players and cooperative AI for team playing games. Our purpose is to develop AI which automatically imitates human like behavior based on Genetic Algorithm and Monte Carlo Tree Search.
We implemented an AI for Mario AI which learns from human play record and evaluated by comparing the output action of the AI and human play record. Our AI achieved 52% match rate and also the actual play of the AI was shown to several cooperator to test whether it is possible to distinguish AI play and human play.

1. はじめに

デジタルゲームプログラミングの国際学会であるCIG (IEEE Symposium on Computational Intelligence and Games)では，毎年様々なデジタルゲームを題材にしたコンペティションが2009年から開催されている．その取り組みの一つとしてSuper Mario Bros.風のゲームのAIの性能を測るMario AI Competitionが開催されている．これまで，様々なアルゴリズムがMario AIを含むリアルタイムゲームに適用されており，強いAIの作成は多くのゲームで可能となっている．

ゲームAIには強さ以外にも重要な性質がある．人間に似たプレイをすることはその一つである．たとえばゲームを開発する際にテストプレイヤーの役割を軽減するAIや，チームプレイの際に実際の人間に代わって協力プレイをするAIなどが考えられる．

本研究ではモンテカルロ木探索法の一種であるUCT (UCB applied to Trees)アルゴリズム[2]と遺伝的アルゴリズム(Genetic Algorithm, GA)を組み合わせた手法を用いることでMario AIを自動的に人間のプレイを模倣するシステムを提案する．UCTはランダムシミュレーション(以下，シミュレーション)をベースとして，シミュレーションによる平均報酬とノードの探索回数から，より見込みのある手に対して多くの探索を行う手法である．Mario AIなどのリアルタイムゲームでは制限時間内に全ての可能性を探索することは困難であるが，シミュレーションを繰り返すことで最良の手を選択できると考えられる．

本稿では実際に構築したMario AIエージェントについての詳細を説明し，GAによる学習方法について述べる．また，数人の被験者に対して作成したエージェントを実際に人間が操作しているように見えるか，アンケートを行った．

[1] 東京大学大学院 総合文化研究科 広域科学先攻 広域システム科学系
Graduate School of Arts and Sciences, The University of Tokyo
東京都目黒区駒場 3-8-1 駒場キャンパス 15 号館 504 室
[2] 湊離散構造処理系プロジェクト
[3] 東京工業大学 大学院情報理工学研究科 数理・計算科学専攻
[a)] nakano.yuk1.hs@gmail.com

2. 背景

2.1 Mario AI と人間を模倣する AI

Super Mario Bros. は 1985 年に任天堂が開発した横スクロールアクションゲームであり，全世界で 4,000 万本を売り上げたコンピュータゲーム史上最も影響力のあるゲームタイトルの 1 つである．このシリーズは主に 1 人でキャラクターを操作するゲームであるが，新作では複数人プレイヤーが協力し同じステージを同時にプレイすることもできる．

2009 年から国際会議 IEEE CIG (Computer Intelligence in Games) において，Super Mario 風のゲームを再現した Java プログラム上で AI 同士が性能を競う，Mario AI 大会が開催されている[*1]．この大会ではステージはランダム生成されているため，先に進むまで状況が分からない．また，Mario AI のゲームはフレーム (1/24 秒) 単位で進むため，毎秒ごとに 24 回行動を選択する必要があり，考慮時間が短いために探索アルゴリズムにとっては厳しい条件である．

大会では，強い (ゲームをクリア能力が高い) AI を作成する競技 (Game Play Track) や，人間の操作に見えるような挙動をする AI を作成する競技 (Turing Test Track)，ステージの自動生成を行う競技 (Level Generation Track) が行われている．

人間のプレイに似た行動をする AI は，人間の代わりにテストプレイを行う AI や，人間とより強調できる AI を作成することに貢献すると期待される．特に First Person Shooter などのジャンルでは需要が大きく，盛んに研究されている．

■ *Mario AI* 特有のルール

Mario AI のルールを示す．

- 探索に用いることができるのは現在までのゲーム画面に表示されている情報のみである．(周囲の地形や敵の場所など)
- AI は 1/24 秒に 1 回 (1 フレームに 1 回)，行動を入力する必要がある．
- 入力はボタンの組み合わせで行い，全 11 通りの行動がある．

■ *Mario AI* のシミュレーション空間

Mario AI のゲーム画面は 19 マス ×19 マスの空間で区切られており，キャラクターの移動もこのマス目を基準に行われる．AI が探索に用いることができる情報は，ゲームに画面現れている操作キャラクターと敵キャラクターの位置，さらにゲーム画面の各マス目の地形である．

2.2 UCT を用いた Mario AI プログラム

人間らしい AI を作成するための元となるアルゴリズムとして主な候補となったものは A* 探索と MCTS であったが，以下の理由により本研究では UCT を元に AI を作成することとした．

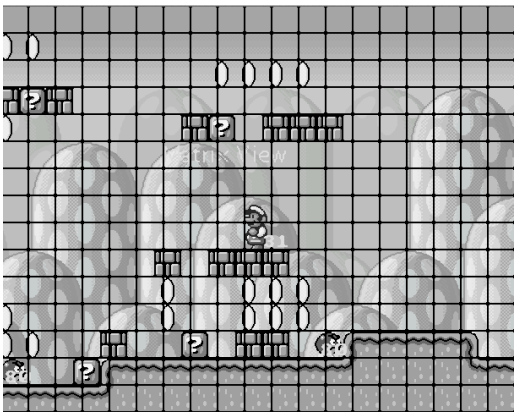

図 1

まず 2009 年の Mario AI Competition で好成績を残した A*探索ベースのアルゴリズム [6] を検証したところ，ステージのクリア時間に関しては人間の上級者比べ同等以上の成績を残した．また，予備実験として UCT アルゴリズムを用いた AI を実装し，FSS2012 で開催された Mario AI 大会[*2]で用いられたステージをプレイさせたところ，同大会に参加した AI と比較して同等以上の成績を残すことを確認できた．

しかし，A* 探索を用いたアルゴリズムの動作は「臆病でない」,「操作ミスをしない」,「短時間でボタン入力が変わる (様に見える)」などの特徴が人間とは異なり振る舞いが非人間的であった．

人間がプレイする際には，通常，操作ミスの可能性などを考慮して，敵を倒すなどして安全なルートを確保してからゴールへ向かう．従って，たとえ最適なルートが発見できても卓越した技術のあるプレイヤー出ない限り敵が多い場所に飛び込み，華麗に敵の攻撃をかわすようなプレイはできない．

人間の持つ慎重さを備えたアルゴリズムを実装するための手法としては，操作上のミスや見積もりの誤差を考え，乱数によって操作に多少の揺らぎを与えた上で，高確率でクリアーできる経路を選択することが考えられる．これは MCTS が行っていることに近いため，UCT は自然と人間のように慎重なプレイをする性質を備えると期待される．

3. 関連研究

3.1 UCT

UCT は多腕バンディッド問題の効率的な解決方法である，UCB1 をゲーム木探索に応用したものである [2].

ゲーム木のノードの展開順序や評価には式 (1) の UCB 値

[*1] http://www.marioai.org/

[*2] ファジイシステムシンポジウム 2012 プラットフォームゲーム AI 大会

により決定される．UCT では UCB 値が高いノードほど，優秀なノードと考えられ優先的に展開される．

$$UCB(i) = \overline{X_i} + C\sqrt{\frac{ln(T)}{Ti}} \quad (1)$$

式 (1) において，$\overline{X_i}$ は i の小ノードの平均報酬，T_i は i 以下の小ノードの全展開数の和，T は総シミュレーション数，C はパラメータである．式 (1) の第 1 項は小ノード報酬の平均を表し，第 2 項は未展開のノードほど値が高くなる．従って，未展開のノードや優秀な候補に多くのシミュレーションが当てられる．

UCT のアルゴリズムは図 2 の通りである．

(1) ゲーム木のルートノードから最も UCB 値の高い小ノードをたどり未展開のノードを決める
(2) 未展開ノードからランダムシミュレーションを行う．
(3) シミュレーションの結果を親ノードをたどりながらに情報を更新する．
(4) 展開したノードが展開数などで一定のしきい値を超えていた場合にノードを木に加える．
(5) 制限時間内まで (i) から (iv) を繰り返す
(6) 最も良いと考えられる手を出力

図 2: UCT アルゴリズム

UCT はランダムシミュレーションを評価値として用いるため，探索に評価関数を必要としない．そのため，新しいゲームや研究対象としての歴史が浅く知識の蓄積が少ないゲームであって，適当な評価関数を予め用意しなくとも自動的に戦略を生成できる．

一般的な UCT では探索時間の最後に，最も探索回数の多かった手を最前の手と考えを出力する．

3.2 リアルタイムゲームにおける UCT

UCT アルゴリズムは囲碁 AI で大きな成功をおさめ，ゲーム以外の問題も含む様々な問題にも適用されつつある探索アルゴリズムである．その特徴は評価関数の代わりにランダムシミュレーション (以下，シミュレーション) を用いることである．通常のゲーム木探索アルゴリズムが苦手とするリアルタイムゲームでも適用例があり，池畑らが開発した UCT を用いたプログラムが 2011 年に Ms. Pac-Man AI Competition で当時の AI の世界記録を達成している [3]. Ms. Pac-Man AI Competition ではゲームと AI のプログラムが独立しており，AI はゲーム画面のスクリーンキャプチャから画像認識を行いゲームの状況を認識する．

Ms. Pac-Man では 1/15 秒に 1 回の入力が必要であり，条件は Mario AI の場合 (1/24 秒に 1 回) と似ている．異なる点としては，Ms. Pac-Man ではゲームのステージが全て画面上に表示されているのに対し，Mario AI ではステージの先の部分は画面外にあるため先に進むまでステージの全ぼうは隠されている点がある．

3.3 進化計算とモンテカルロ木探索

モンテカルロ木探索ではシミュレーション部分の性質がアルゴリズム全体の性質 (強さを含む) に大きな影響を与える．しかしどのようにシミュレーション部分を作成すれば強くなるのか，はっきりしたことは明らかになっていない．

シミュレーション部分の調整に進化計算を用いることは有効である可能性がある．MCTS のシミュレーション部分の強化を遺伝的プログラミングを用いて行う手法が，2013 年に発表されている．Benbassat らはオセロなどに対して [4]，Alhejali らは Ms. Pac-man AI に対して [5] それぞれ適用している．

4. 提案手法

本研究では人間が操作している様なプレイをする Mario AI エージェントを作成するための手法として，UCT を基本とし，さらにランダムシミュレーションとその報酬を GA を用いて調整する手法を提案する．本章ではその手法について述べる．

(1) MarioAI から画面の情報を取得
(2) 画面の情報から報酬体系，ランダムシミュレーションの方策を決定
(3) 制限時間まで UCT アルゴリズムを実行
(4) 制限時間が来たら最もよいと考えられる行動を出力

図 3: Mario AI における UCT

4.1 ゲーム画面から得る局面の特徴

本研究では UCT アルゴリズムの報酬やシミュレーション時の行動選択の確率に対するパラメータを設定した．人間のプレイ履歴を教師例としてこれらの値を調整することによって，AI の選択する行動と人間のプレイ履歴の一致率を高めることを試みた．

Mario AI における操作キャラクターは敵キャラクターと接触するとダメージを受けてしまうため，図 4 のように敵が近い (赤)，敵が遠い (青)，または敵が全くいないような場合ではそれぞれ優先すべき行動やシミュレーションの報酬を変更するべきである．さらに，図 5 のように目の前に段差がある際には，キャラクターはジャンプをしなければ段差を超えることができずに前に進む事ができないため，

優先的にジャンプを含む行動を探索しなければならない．人間のプレイヤーがゲームをプレイする際にも，これらの情報を考慮し最も有効であると考えられる行動を選択すると思われる．

今回はステージの特徴として**敵が近い**，**敵が遠い**，**段差がある**，**fireball を打てるかどうか**，**jump 入力が有効かどうか**を用いた，これらが成立しているかどうかを {0,1} として，この配列を特徴ベクトルと定義する．特徴ベクトルは 1 フレームごとに更新され，その都度値が変化する．

以上の特徴に対して，UCT のシミュレーションにおける方策に関わるもの (行動の重みベクトル) と，シミュレーションの報酬に関わるもの (報酬の重みベクトル) の 2 種類のパラメータを設定した．

本研究ではこれらのパラメータを GA により調整した．

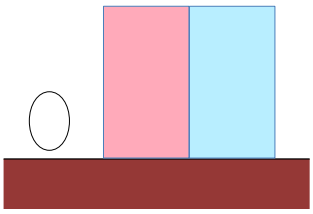

図 4: 敵との距離

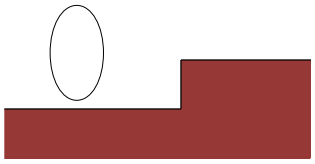

図 5: 段差

4.2 ランダムシミュレーションの方策

UCT ではシミュレーションの方策が探索の性能に大きな影響を与える．囲碁などにおける UCT でのランダムシミュレーションでは人間のプロ棋士の棋譜などから学習したデータから，頻出するパターンに対しては優先的に探索する手法がとられている．

Mario AI での UCT のシミュレーション時の行動選択に関して，全ての行動から等確率にランダムに行動を選択してしまうと，シミュレーション最中に図 6 の 11 通りの行動[*3]を行う二次元平面上のランダムウォークになってしまうような状態が観察された．従って，シミュレーション時の行動選択の際に (ゲームの状態によって変化する) 特定の行動を優先させる機能を実装した．

実際の探索では 1 フレームの間に使える探索時間は 42 ミリ秒であるため，シミュレーションの長さは 24 フレーム[*4]と制限した．また，この条件では 1 フレームあたりに 300 回程度のシミュレーションを実行できる．

■ 行動の重み

UCT ではシミュレーションの過程で確率的に行動を選ぶが，探索時のフレームの特徴次第では優先して探索すべき行動が変化する．

[*3] 図は「しゃがむ動作」など一部簡略
[*4] ゲーム内では 1 秒間にあたる

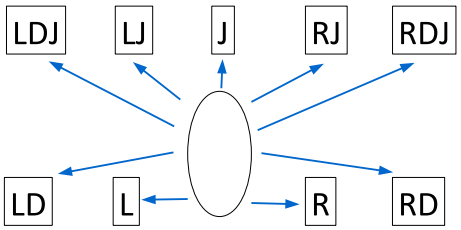

図 6: Mario AI の行動

R	右
L	左
D	走る
J	ジャンプ
RD	右に走る
LD	左に走る
RJ	右にジャンプ
LJ	左にジャンプ
RDJ	右に走りながらジャンプ
LDJ	左に走りながらジャンプ
C	しゃがむ

表 1: 11 種類の行動

Mario AI のようなゲームのプレイ動画は様々な動画サイトに世界中の人がアップロードをしているため，それらの動画からプレイヤーの操作を検証した．その結果，特定のステージの状態になると優先して行われる操作として，敵が近いと攻撃ボタン，段差があるとジャンプボタンなどを連打しているプレイヤーが多いことが判明した．従って，ゲームの状態によって人間が優先して入力する操作があると考え，操作キャタクターの全てのアクションに重みを定義して各アクションを探索する割合を制御した．

Mario AI のキャラクターの操作はボタンの入力の組み合わせから成り立っている．キャラクターを操作するボタンは { R , L , D , J , C } の 5 種類であり，有効な入力の組み合わせは表 1 の 11 種類である．そのため，ゲーム操作のためのの全てのボタンに重みを設定した．

■ 実装方法

各ボタンに特徴ベクトルと同じ要素数を持つベクトルを用意し，「ボタンの重みベクトル」とする．ボタンの重みベクトルの要素は非負の実数とする．

$$w_i (i \in \{R, L, D, J, C\}) \tag{2}$$

w_i はボタン i の重みベクトルである．ボタンの重みベクトルと，4.1 章の特徴の重みベクトル F との内積 b_i を計算し，これをボタンの重みと呼ぶ．

$$b_i = w_i \cdot F \tag{3}$$

ボタンの重み b_i は各ボタンを入力する割合を決めるパラメータであり，探索においては UCT アルゴリズムを実行する度にステージの情報を用いて計算される．実際の入力はボタンの組み合わせから成り立っているため，行動の入力の割合を以下の式で決定する．

1 つのボタンのみの入力の場合．

$$P_i = a_i \ (i \in \{R, L, D, J, C\}) \tag{4}$$

複数のボタンを入力する場合は，入力している全てのボタンの重みの積を用いる．例えば RDJ なら以下の式である．

$$P_{RDJ} = a_r \cdot a_d \cdot a_j \tag{5}$$

ランダムシミュレーションの確率的な行動選択には式 4，式 5 で計算される値をもとに，全行動からルーレット選択により行動を決定する．重みの初期値は 1.0 とし，式 5 が 1.0 より大きければ優先度が高く，0 に近ければ優先度が低いことを示す．

4.3 報酬

UCT では探索をする際にゲーム局面の評価関数を用いるのではなく，ランダムシミュレーションの報酬を元に最善と考えられる手を決定している．従って，シミュレーションの報酬が探索の方針に大きな影響を与える．囲碁などのようなゲームではランダムシミュレーションをゲームの終局まで行い，勝敗の結果を報酬としている．しかし，Mario AI の様な不完全情報ゲームではゲーム画面に映っている情報のみを用いて探索を行う必要がある．そのため，ランダムシミュレーションではゲームのゴールには到達できず，勝敗などの情報を直接得ることはできない．

本研究ではシミュレーション前後でのキャラクター状態の変化を用いて評価を行った．

■ 報酬に用いた要素

今回，UCT のシミュレーションの報酬は水平，垂直方向への移動距離や敵の討伐数，敵から受けたダメージなどを用いて決定した．これらの要素の差分はシミュレーション終了後に容易に取得でき，各要素ごとに重みをかけ正規化したものを用いて報酬体系を自動的に作成した．

Mario AI のような横スクロールゲームでは図 7 のような袋小路上の構造が随所にみられる．このような構造に侵入してしまうと，水平方向の値ではゴール側に近づいている用に見えるが，実際に画面右端に到達するには一度，画面左方向に迂回する必要がある．過去の大会に参加したエージェントもこのような構造に侵入してしまうと抜け出せない様な状態が見られた．そのため，袋小路のような構造を抜けるために，画面右端をシミュレーションのゴールと考え，図 7 のようにゲーム画面を格子状に分割し，画面右端の格子の値を 0 とし，隣接する格子に移動するごとに値が 1 増える距離空間を定義した．報酬にはシミュレーション前後のこの距離の増減も用いた．以下，この距離をプラットフォームゲーム離散化距離 (PDL) と呼ぶ．

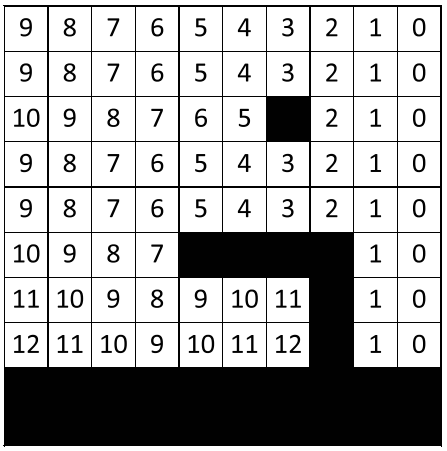

図 7: 右端からの距離 (PDL)

■ 報酬の重み

Mario AI では UCT のシミュレーションの報酬の設定により AI の性格を臆病や攻撃性などに変える事ができる．人間のプレイに似せるためには単に最適なルートを検索するのではなく，為には単に最短路を探すような報酬ではなく，最適でないような行動を選択する必要もある．

従って，人間がプレイする際に重視する要素を考え，**敵をした数**，**水平移動距離**，**垂直移動距離**，**図 7 の PDL**，**受けたダメージ**を報酬の計算に利用する (表 2)．

探索時のステージの状態次第では，報酬の要素はのうち，重視すべき要素が異なる．従って，4.1 章の特徴ベクトルから最適な報酬体系を作成するため，表 2 の要素に対してパラメータを設定した．

■ 実装方法

表 2 の要素に対して，長さ 5 の「報酬ベクトル」を用意する．報酬ベクトルは非負の実数値を要素として持つ．

$$w_j \quad (j \in \{dx, dy, dpdl, kill, damage\}) \tag{6}$$

報酬ベクトル w_i と特徴ベクトル F の内積を計算し，これを報酬の重みとする．

$$reward_j = w_j \cdot F \quad (j \in \{dx, dy, dpdl, kill, damage\}) \tag{7}$$

報酬の重み $reward_j$ 報酬の各要素の重要度を表す．シ

ミュレーションの報酬 REWARD は，$reward_j$ と報酬の要素 j の積の総和により報酬を定める．

$$REWARD = \sum_{k \in \{dx,dy,dpdl,kill,damage\}} reward_k * k \quad (8)$$

dx	水平移動距離
dy	垂直移動距離
dpdl	PDL の移動距離
kill	敵を倒した数
damage	受けたダメージ

表 2: 報酬の要素

4.4 人間に似せるための制限

今回，通常の UCT の実装を行ったところ，そのエージェントは 1 フレームごとに行動を変化させるような出力を行った．しかし，人間のプレイでは，実際に指でボタンを押すことによりゲームのキャラクターに指示を与えているため，熟練のゲーマーではない限り AI のように 1 フレーム (42 ミリ秒) ごとに入力を変更することはできない．

本研究では，AI の選択した行動が人の操作のように見えない原因を，出力が短い間隔で変化することと考えた．そのために，ランダムシミュレーションの行動選択とアルゴリズム全体の行動選択の出力に関して，毎フレーム行動を変化させるのではなく数フレームに一度のみ行動を変化させる制限を加えて実験を行った．

4.4.1 出力に関する制限

人間と AI のプレイ履歴を比較したところ，人間による操作ではあるボタンの組み合わせが入力されると数フレームの間は同じボタンの組み合わせが入力がされている事ことがわかった．従って，UCT の出力を制限することにより人間らしいプレイをするエージェントの作成を目指した．

エージェントの出力を制限するために，探索を毎フレーム実行するのではなく一定のフレーム間隔に一度だけ探索を行い，探索をしてない間は前回の探索の結果を引き続き入力し続けた．この数フレームの間隔を**同一行動保持時間**と呼ぶ．

4.4.2 シミュレーション中の制限

UCT ではランダムシミュレーション結果から局面を評価しているため，シミュレーションの行動選択が探索の方針に大きな影響力を持つ．そのため，シミュレーション中も人間の操作のような行動を選択をする必要があると考え，シミュレーション中の確率的な行動の変更も数フレームに一回と制限した．

また，同様にシミュレーション中にも人間に近い動作を行わせるために，**行動変更の間隔**を定義し，シミュレーション中でも行動変更の間隔に一度だけ，行動の変化をすることとした．

5. GA による学習

5.1 学習方法

今回，GA によって人間のプレイを学習することによって人間のプレイに似た操作をするエージェントを作成した．人間のプレイ履歴の全フレームに対して UCT を実行し，その出力が人間の操作と一致した率を GA の評価基準として学習を行った．学習するパラメータはパラメータは行動の重みベクトル，報酬の重みベクトルとした．

■ 個体の表現

個体の表現方法として，行動の重みベクトル，報酬の重みベクトルを連結した一次元の配列を用いた．この配列は長さが 50 で非負の実数からなる．

■ 交叉

個体の交差は，2 個体のパラメータの一次元の配列に対して n 点交叉を行った．n 点交叉ではパラメータ配列から任意の n カ所を選出し交叉を行う．

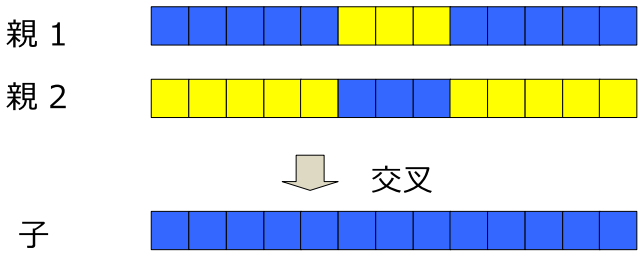

図 8: 2 点交叉の例

■ 突然変異

個体のパラメータに対して以下のように突然変異を適用した．パラメータの各要素を一定の確率 P_m で変異させた．変異の範囲は変異幅 c を設定し，c 倍から $1/c$ 倍の範囲とした．[*5]

■ エリートの選択

各世代ごとに評価値の高い E 個のエリート個体を残すようにした．

■ 評価値

本研究では人間がプレイした履歴から学習し，人間の操作のように見えるエージェントの作成を目的とするため，人間のプレイ履歴との一致率を評価値として用いた．一致率は，学習対象のプレイ履歴の全フレームに対して UCT を実行し，それぞれのフレームで選択された行動が人間のプレイ履歴と一致したフレームの割合と定めた．

[*5] 実装では $1/c$ から c の一様乱数としたため値が大きくなる傾向があった．

5.2 学習方法

本実験で学習に用いたサンプルは，人間の 60 秒間のプレイ履歴で，$60*24(=1440)$ フレームから成り立っている．

5.3 手順

パラメータの調整を図 9 の手順で行った．

(1) ランダムに初期世代を作成
(2) 個体ごとのパラメータ設定で UCT を行い一致率を評価．
(3) “ 交叉 ”, “ 突然変異 ”, “ エリートの選択 ” により次世代を作成．
(4) (2) から (3) 繰り返し

図 9: GA による学習

5.4 個体の初期化

個体の初期化方法はランダムに決定した．パラメータの総和 W を決定し，50 個のパラメータに対してランダムに分配した．

6. 性能評価

今回作成したエージェントの評価基準として人間が操作したプレイ履歴との一致率を用いた．

実験に際してエージェントは 4.4 章で提案した「同一行動保持時間」,「行動変更の間隔 」の設定に対して，それぞれを有効，無効にした 4 つの UCT エージェント（表 3）を作成し実験を行った[*6]．

エージェント	出力の 同一行動保持時間	シミュレーション中 の行動変更の間隔
A	1 フレーム	1 フレーム
B	1 フレーム	5 フレーム
C	5 フレーム	1 フレーム
D	5 フレーム	5 フレーム

表 3: 実験に使用したエージェント

6.1 GA による学習

5 章で述べた GA による学習方法でパラメータを調整した．実験には Intel Xeon CPU 2.3GHz, メモリ 16GB のマシンを用いた．

また，実験に用いた GA の設定は表 4 の通りである．初期値，エリート数，突然変異確率，変異の幅に関して表 4 の 16 通りの設定で GA を実行し，各エージェントの一致率を高めた．

個体数	30	初期値 W	35 , 50
世代数	30	エリート数 E	1 , 3
交叉確率	0.7	突然変異確率 P_m	0.5 ,0.1
n 点交叉	4	変異の幅 c	5 , 20

表 4: GA のパラメータ

GA によって各エージェントが最良の一致率を残した際の結果を図 10，最終世代の平均値と最良の個体を表 5 に示す．

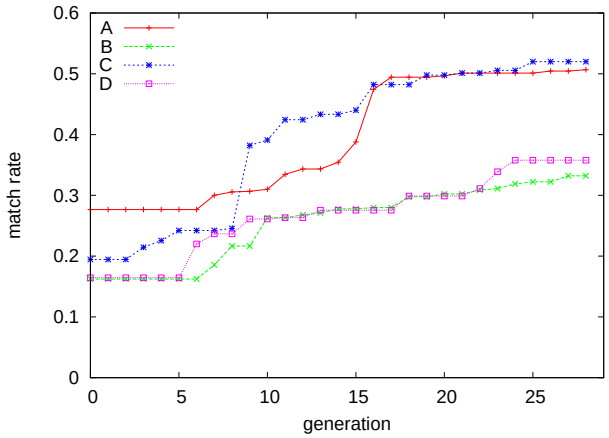

図 10: GA による学習の結果

	A	B	C	D
mean	0.448	0.298	0.453	0.305
max	0.507	0.332	0.520	0.358

表 5: 最終世代の一致率の平均値と最大値

実験の結果，次の設定 $(W,E,P_m,c)=(35,3,0.5,20)$ で 4 エージェントともに最も高い一致率を残した．ここでは最善の結果のみを示した。特にエージェント A，C に関してはともに 50%以上の一致率を達成している。最終世代の平均値に関しても A，C は他の二つに比べ高い値を示しているため，人間の操作履歴から十分に学習できたと考えられる．また，4.4 章の設定に関して比べると，シミュレーション中の行動変更の間隔が 1 フレームである A，C の一致率は良いが 5 フレームである B，D では一致率が悪いことがわかる．また、出力の同一行動保持時間が 1 フレーム，5 フレームのどちらの設定でも，一致率には大きな差はなかった．

[*6] 学習のサンプルに用いた行動履歴から 5 フレームを妥当な長さと判断した．

6.2 アンケートによる評価

さらに各エージェントの評価のためにアンケートを行った．CIG の Mario AI Competition で行われている Turing Test Track ではオーディエンスによる投票によって，エージェントの評価をしており、その方法を参考とした。

6.2.1 アンケートの方法

本研究で提案した手法を取り入れたエージェントのうち、明らかに挙動が人間らしくない[*7]A を除いた 3 つ、B, C, D と、学習の対象として用いた人間によるプレイ (以下，H とする) の動画[*8]を作成し，参加者には人間のプレイらしく見える順番にエージェントに順位を付けてもらうよう依頼した．

アンケートの参加者は本研究のアルゴリズムについての説明や解説は受けておらず，どのような手法でエージェントが探索を行っているかは知らされていない．また，本研究で目標とした人間らしいプレイに必要な要素についても知らされていない．

6.2.2 アンケートの結果

アンケートの結果としてエージェントの順位を示す．表 6 では，エージェント X がエージェント Y より人間のプレイのように見える状態を「$X > Y$」と表記した．

参加者 1	$H > C > B > D$
参加者 2	$H > C > B > D$
参加者 3	$C > B > H > D$
参加者 4	$B > C > D > H$

表 6: アンケートの順位

アンケートの順位よりエージェント C の評価が高く，エージェント D が最も評価が低いことがわかる．

エージェント C に対するコメントとして「人間の操作と考えても問題無い」や「中級程度の人間の操作」などが聞かれた．また，エージェント B に対しては「行動の切り替えが早すぎるため，人間ではない」,「操作が上手すぎる」となどの意見があった．エージェント D に対しては「頭の悪い AI」や「目的がわからない行動が多い」と行った評価を受けた．

7. おわりに

我々は UCT と遺伝的アルゴリズムを組み合わせ，Mario AI において人間のプレイ履歴を元に人間のプレイを模倣する AI を作成した．それだけではプレイ履歴に対する一致率を高めることは可能だったが，動作が不自然だったために同一行動保持時間という制約を加えて実験を行った．

数名の被験者に対するアンケートからは人間のプレイと誤解されることもあったことが分かる．ある程度人間らしい AI を作成できたため，提案した手法は Mario AI という比較的単純なゲームでは有望だと言える．

しかし学習に用いた特徴が少なくかつ単純なものであり，手法自体がまだ洗練されていないなど，改良の余地が大きい．また，今回は人間のプレイを模倣することを目的としたが，本手法を，例えば単純に性能の高い AI を作成するなどの他の目的に応用することも検討したい．これらは今後の研究課題である．

謝辞 本研究は JSPS 科研費 23700158, 23240005, 25700038 の助成を受けたものです。

[*7] 出力などに対して制限を設けていないため，行動が 1 フレームごとに変化してしまい人間のプレイらしくない．

[*8] ステージは同一のものを用い，動画の長さは 1 分程度．

参考文献

[1] Finite time analysis of the multiarmed bandit problem. P. Auer, N. Cesa-Bianci and P. Fischer. Machine Learning, Vol. 47 (2–3), pp. 235–256, 2002.

[2] Bandit Based Monte-Carlo Planning. L. Kocsis and C. Szepesvári, 17th European Conf. on Machine Learning (ECML 2006), pp. 282–293, 2006.

[3] Monte-Carlo tree search in Ms. Pac-Man. N. Ikehata and T. Ito, IEEE 2011 Conference on Computational Intelligence and Games (CIG 2011), pp. 39–46, 2011.

[4] EvoMCTS: Enhancing MCTS-Based Players through Genetic Programming. A. Benbassat and S. Moshe, IEEE 2013 Conference on Computational Intelligence and Games (CIG 2013), 2013.

[5] Using Genetic Programming to Evolve Heuristics for a Monte-Carlo Tree Search Ms Pac-Man Agent. A. Alhejali and S. Lucas, IEEE 2013 Conference on Computational Intelligence and Games (CIG 2013), 2013.

[6] The 2009 Mario AI Competition. J. Togelius, S. Karakovskiy and R. Baumgarten. IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, 2010.