# 横スクロールアクションゲームにおける多様なスピードランルートの自動提案

原口海[†,1]　池田心[†,2]

**概要**：スピードランとは，主にアクションゲームをできるだけ短い時間でクリアする“遊び方”であり，日本ではタイムアタックなどとも呼ばれて親しまれている．スピードランをうまく行うためには，適切なルートつまりステージの進め方を設定したうえで，それを正確にこなす必要がある．上級者のルートは動画サイトなどで見つけられるが，それは初中級者には模倣が難しいものである．そこで本研究では，Mario について初中級者でもプレイできるようなルートを自動的に提案するようなシステムを目指し，これをペナルティ付き遺伝的アルゴリズム(GA)で実現することを目指した．最速クリアのみを目指した GA では 12 回も敵キャラとニアミスをしたが，ニアミスにペナルティを加えた GA ではそれを 4 回に抑えることができた．面白い点として，後者のほうがクリアタイム自体も早くなったことも挙げられる．

**キーワード**：スピードラン，ルート生成，遺伝的アルゴリズム，ペナルティ

# Evolution of Various Speedrun Routes for Platform Games

Kai HARAGUCHI[†,1]　Kokolo IKEDA[†,2]

***Abstract***: "Speedrun" (called "time attack" in Japan) is a way of playing action games, where each player tries to clear the given stage/game as fast as possible. For this purpose, each player should optimize his route to play from the start to the goal, and also should play the route accurately. Good routes optimized by experts can be easily found from the Internet, but such routes will be too difficult for common players. So, this research aims to generate and suggest good routes for common players, and proposes a way to realize the suggestion system by using a genetic algorithm (GA) with penalties. We compared two routes, one by a GA for fastest clear and the other by a GA for fast and safe clear. While 12 dangerous situations were found in the former route, the number was successfully decreased to 4 in the latter route. One more interesting point is that the clear time of the latter route is faster than that of the former route.

***Keywords***: Speedrun, Route generation, Genetic Algorithm, Penalty

## 1. はじめに

近年，深層学習の登場や計算機技術の向上により，人工知能を用いた研究は幅広い分野で行われている．ゲームの分野では，人間では困難なステージをクリアするアクションゲームの AI[1]や, 人間チャンピオンに打ち勝つ囲碁 AI[2]などが開発され，“クリア”や“勝利”といったゲームの主目標を達成する AI は完成しつつある．近年では，人間らしい動きを模倣したり [3], 教師となったりする研究が盛んに行われており，AI の役割は協力者や教育者へとシフトしている．しかし，これらの研究においても最終的な目標は人間プレイヤーの“クリア”や“勝利”の補助である場合が多い．

これらとは異なり本研究では，ユーザーによって“後から追加された目標”に着目する．具体的には，横スクロールアクションゲームにおいて，ただステージをクリアするのではなく，できるだけ素早くクリアすることを目標とする「スピードラン（タイムアタック）」という遊び方をとり

† 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, 923-1211, Japan
1 s1810149@jaist.ac.jp
2 kokolo@jaist.ac.jp

あげる．詳細は次章で述べるが，この目標を AI がサポートするためには，プレイヤーの技術レベルに応じたスピードランルートの設定というものが必要であり，本研究では遺伝的アルゴリズム（GA）でこれを行う． 対象のゲームとしてはスーパーマリオを用いる．研究が盛んに行われている，評価が行いやすい，プラットフォーム[4]が公開されている，などの理由からである．詳細は 3 章で述べる．

## 2. スピードランについて

本研究で設定した“素早いステージクリア”という目標は，スピードランやタイムアタックと呼ばれる，やりこみ要素の一種である．近年では動画投稿サイトなどで人気のプレイスタイルであり，e-sports として大会[5]も開催されている．やりこみ要素ではあるものの，超上級者でなければ挑戦できないようなものではなく，小学生であれ中高年であれ自分が達成可能な目標を定めて，スキル向上や隣人との競争を楽しむことができる，懐の広い世界である．

競技会でのルールはさまざまで，対象となるゲームやステージはもとより，「最終ボスを倒すまで」や「隠しアイテムを全回収するまで」などの達成すべき目標が定められている．また，バグ利用の可否なども定められることが多く，いずれにせよそれぞれのルールに沿った攻略法が必要とされている．

スピードランにおける時間短縮を意識したクリアまでの道筋（ステージの進め方）は，“ルート”や“チャート”と呼ばれ，良いルートの発見はタイム短縮の大きな役割を担っている．自分に合った良いルートを発見することは簡単ではないため，それ自体が長期的な意味でのスピードランの楽しみ方のひとつでもある．

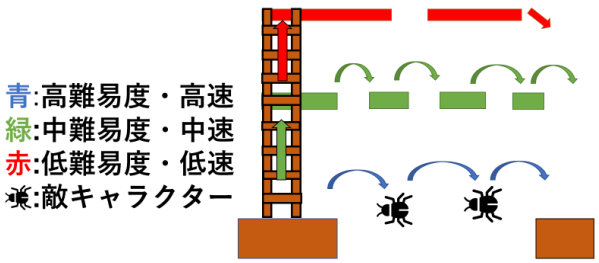


図 1　スキルによるルートの違い

例えば，図 1 に示したような横スクロールアクションゲームにおけるスピードランでは，プレイスキルに応じたルートの選択が重要である．最も早いルートは青で示した，敵キャラクターを足場にして対岸に渡るルートだが，失敗した場合はリカバリーが効かないためリスクが非常に高い．一方，赤のルートは安全だが，その分時間がかかる．

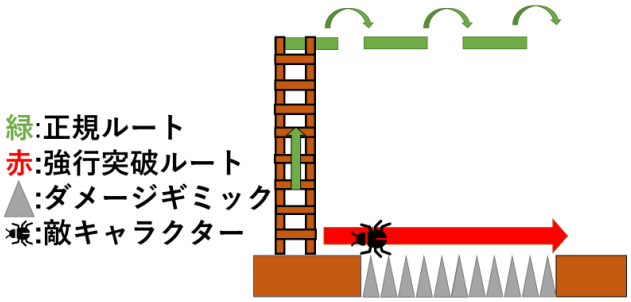


図 2　ギミックを無視した強行突破の例

また，図 2　ギミックを無視した強行突破の例に示すような，本来であれば遠回りする場面を，敵と接触した際の無敵時間を利用した強引な突破法も度々取られる．強行突破自体は容易に行えることも多いが，これを実行する“ルート”も同様に簡単とは限らない．このルートでは確実に負傷するため，図 2 のポイントまで体力を温存して到着する必要があるうえ，途中で無敵が解除されれば再度負傷する．この前後にボス戦があれば，回避に専念することで短縮分以上の時間を失うこともある．

このようにスピードランでは，瞬間的に最善の手が長期的にも最善であるとは限らない．スピードラン上級者であれば自身のスキルとリスクを照らし合わせて選択できるが，スピードラン初級者にとっては自身に適切なルートの選択は困難である．それにも関わらずルート探索には多くの経験やゲームそのものに対する知識が必要であるため，多くのプレイヤーは動画投稿サイトにアップロードされたものを模倣して獲得している．しかし，それらの動画はトッププレイヤーがプレイするものがほとんどであるため，初心者が自身に合ったルートを発見することは難しい．また，動画上では一見簡単そうに見えても，コンマ 1 秒以下の入力速度を要したり，小さなミスが大幅な遅滞に繋がるハイリスク・ハイリターンなものだったりする． このようなルートを初中級者が真似を試みても失敗ばかりで面白くなく，技術向上の妨げにすらなりかねない．

そこで本研究ではスピードランにおけるルート探索を AI に補助させ，より多くの選択肢の提供を試みる．通常の「強い AI」「人間らしい AI」では，“未知の”ステージに対して良いプレイを行う AI を，A*アルゴリズム・強化学習・入出力モデル＋遺伝的アルゴリズムで最適化することがよく見られる．しかし本研究の対象であるスピードランは，通常対象ステージが事前に公開されている．そのため，“1 つの”ステージに対する良いプレイを行えれば十分である．この点を鑑みて，本研究では遺伝的アルゴリズムでルートを直接探索することを試みる．これにより，探索の際コマンドの入力速度や敵とすれ違う際の距離などに制限を加えることで，幅広いスキルレベルに対応した多様なルートの提案

も期待できる．

## 3. 関連研究

スーパーマリオは極めて有名なゲームであり，一人ゲームで評価が行いやすいことや優れた研究用プラットフォームが早くから公開されたことから，さまざまな研究が行われている．高速なクリアを目指すもの，人間らしい動きを目指すもの，人間のレベルに合わせたステージを生成するものなどであり，IEEE CIG などの国際会議でも多様な競技会が行われてきた．

高速なクリアを目指すものとしては，A*アルゴリズムによるもの[1]がその動画のインパクトもあって有名である．基本的には，右側に進めば進むほどゴールに近くなるというヒューリスティック関数を与えることによって探索を効率化しており，S 字の動きなど迂回を要しないステージでは極めて高速なクリアが可能である．しかしながら，これをスピードランのルート設定に用いることは適切ではないだろう．1 ピクセルレベルでの敵とのすれ違いや，極めて厳密な操作が必要となるためである．なお A*はオンラインのアルゴリズムであり，オフラインでの学習は必要としない．

事前の学習を必要とするアプローチとしては，遺伝的アルゴリズムや強化学習がしばしば用いられる．Togelius らは Neuroevolution という枠組みを用いてマリオのプレイヤーを進化させている[6]．これはマリオの周辺状態を入力とし，行動を出力とするニューラルネットワークを最適化させる遺伝的アルゴリズムであるが，ポイントとしてネットワークの結合重みだけでなく構造も変化させる点がある．A*と同様この方法は未知のステージにも適用できるが，学習に用いたステージでの性能よりも落ちるのは良く知られたことである．

藤井らは，人間らしいマリオの AI を学習させるための取り組みを行った[3]．マリオに限らず多くのゲームに普遍的に登場する人間らしさである“身体的制約”に注目し，状態認知に遅れやあいまいさを入れる，操作に疲れの要素を入れるなどの提案を行っている．強化学習および A*アルゴリズムをベースとして，ここに提案手法を加えることで，人間よりも人間らしく見えるようなエージェントの獲得に成功している．

注意すべき点は，ここで用いられている手法または獲得された政策は基本的には「多くのステージで利用可能なもの」を目指しているということである．

## 4. 提案手法

前章で紹介した手法は基本的に全て未知の状況における AI エージェントの挙動を良くしようというものである．これに比べ，スピードランでは対象ステージが事前に分かることが多いことを踏まえ，本研究では「与えられたそのステージの中で，適切に行動する方法（良いルート）が分かればよい」と割り切る．そして，（汎用的な戦略ではなく）ルート自体を直接的に最適化することを目指す．その場合でも，ルートをどのように表すかにはいくつかの可能性があり，これは 4.1 節で述べる．4.2 節と 4.3 節ではルートの通常の評価方法，4.4 節ではさらに“プレイしやすいルート”のための評価方法を述べる．実際にルートを最適化する方法については 5 章で述べる．

### 4.1 スピードランルートの直接探索

本研究では，行動をルートを構成する要素と考えることで，スピードランルートの直接探索を試みる．

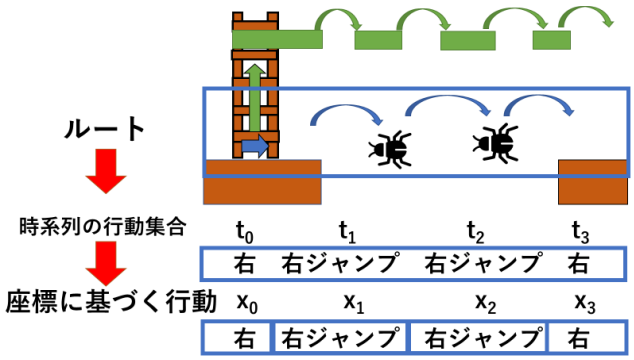


図 3　ルートの直接探索手法のイメージ

図 3 のルートはジャンプで敵を踏んで対岸に渡るルートを示している．このルートの達成には，右→右ジャンプ→右ジャンプ→右という入力を時系列順に行う必要である．しかし，ルートを時系列に基づく行動の集合と捉えた場合，探索が非常に困難となる．例えば，$t_0$ の「右」という行動が「上」に更新された場合，いったんは梯子を登るが $t_1$ で「右にジャンプ」してしまい落下してしまう．これを防ぐには，梯子を上がりきるまで「上」を押すなど，$t_1$ 以降の全ての要素に大幅な変更が必要となる．

一方，ルートを座標に基づく行動の集合と捉えた場合，場合，$x_0$ が「上」に更新されても影響は小さく済む．x 座標が変わらないため，梯子を登り終わった後も上を目指し続けるが，後ろの要素は影響を受けない．座標が固定された場合への対処は必要となるが，更新が要素全体に及ぶことと比較するとその労力は著しく減少する．

本研究では以下の手法を用いてルートを行動に分解し，行動の更新することでルートの探索を行っている．

- ルートを「この x 座標の範囲ではこの行動を取る」と表現することにする．
- x 座標は連続的であるため，全ステージをある程度幅の狭い N 個のブロックに分けることにする．
- 1 つのブロックごとに 1 つの行動，全部で長さ N の行動の配列でルートが表現できる．
- エージェントによるテストプレイでルートの“良さ”を評価する．例えばステージのどこまで進めるかなど

である．

- 長さ N の配列を 1 つの個体とした，遺伝的アルゴリズムによって，評価値が高くなるようなルートを最適化する．

なお，ステージによっては右→上→左→上→右のように「同じ x 座標でも違う行動を取らないといけない」ことがありうる．この場合には，y 座標についても分割を行う必要があるだろう．

### 4.2 エージェントによるテストプレイ

上記の通り本研究では，ある座標 x における行動を更新してルートの探索を行っているため，評価にはエージェントによるテストプレイが不可欠である．Mario AI Benchmark[4]は，横スクロールアクションゲームを対象としたAIのベンチマークソフトとしてしばしば用いられている．図 4 に示したように，ステージ生成や，x 軸座標などの環境情報の習得が容易に行えるため，本研究でもこの環境を利用してテストプレイを行う．

このベンチマークソフトのエージェントの有効な行動は表 1 に示す 12 種類である．

表 1　エージェントが取り得る行動一覧

| | |
|---|---|
| 無操作 | 右＋ダッシュ |
| 右 | 左＋ダッシュ |
| 左 | 右＋ジャンプ |
| しゃがみ | 左＋ジャンプ |
| ジャンプ | 右＋ジャンプ＋ダッシュ |
| ダッシュ | 左＋ジャンプ＋ダッシュ |

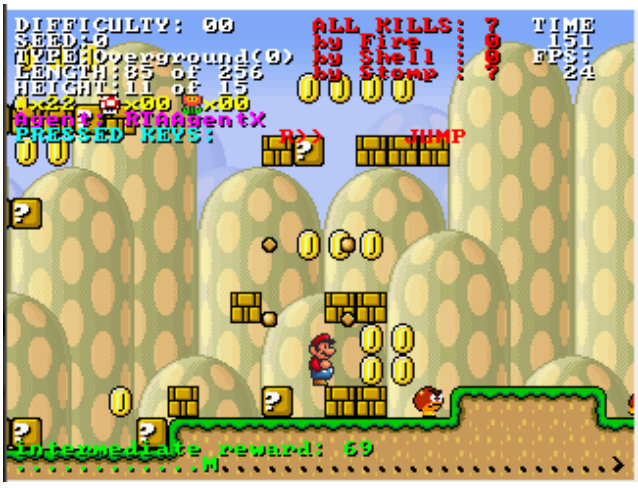


図 4　Mario AI Benchmark の画面

Mario AI Benchmark を用いる研究には，周辺の状況などの“入力”をもとにして次の行動を定めるものが多い．しかし，本研究で利用するのは x 座標のみであり，4.1 節で説明した長さ N の配列を参照して行動が簡単に決まっていく．具体的には，“x 座標を取得→当該インデックスの行動を 5 フレーム実施”を繰り返すことが基本となっている．

行動を 5 フレーム連続して実施する理由は，“その行動をプレイヤーが実施できるか”を考慮したものである．Mario AI Benchmark の意思決定タイミングは 1 秒間に 24 回（24FPS）であるが，0.04 秒ごとに行動を変えるなどということは普通の人間には不可能である．そこで，切り替え回数を軽減するためにこのような措置を取った．

また，1 秒以上マリオの x 座標に変化がない場合は，強制的に右＋ジャンプを実行することで行き詰まりを緩和している．横スクロールアクションゲームでは左端にスタートがある場合がほとんどのため，スタートから遠ざかる右側を選択する．同時にジャンプを行うことで障害物に引っかかった場合の対処も兼ねる．このような処理を行わなくとも綺麗にクリアできるルートは表現できるかもしれないが，これがなければランダムに作った解はすぐに停止または障害物にひっかかってしまって探索効率が悪かったため，用いることにした．

### 4.3 生成されたルートの評価

本研究において，生成されたルートの評価はエージェントを評価することで行う．エージェントの評価には Mario AI Benchmark のスコアを利用する．デフォルトでは敵キャラクターのキル数や獲得したコインやアイテムの数で計算されるが，スピードランにおいてはこれらの値は（それが求められる特殊な競技ルールの場合を除いて）不要である．したがって，どれだけ進んだか，クリアできたか，クリア時にどれだけ時間が残っていたか，を主に参照することにする．

本実験ではエージェントがステージクリアに成功した場合（数式 1）と失敗した場合（数式 2）の 2 つに評価関数を分けて，評価値の計算を行った．

$$E_{clear} = distance + 1024 + 10 \times leftTime \quad - penalty \quad (1)$$

$$E_{failed} = distance \quad - penalty \quad (2)$$

*distance*, *leftTime* はそれぞれで示したスコアの到達距離，残り時間が代入される．ゴールに到達できなかった場合には distance から penalty を差し引いた値のみが評価値として計算される．

### 4.4 penalty

単純に進んだ距離や残り時間を最大化した場合，得られたルートには「プレイしやすさ」といった本来欲しい特徴が考慮されていない可能性が高い．例えば，敵のぎりぎり近くをすり抜けるルート，足場があるぎりぎりでジャンプ・着地するようなルート，動作の切り替えが頻繁にあるルートなどは，一般的に初中級者向きとは言えない．

こういった好ましくない特徴を, if-then ルールなどでハードに禁止することはできるだろうが，それでは今度は主たる性能（クリアの速さ）が落ちてしまうこともありうる．こういったときに，ペナルティという形でこれをソフトに抑制することができるのは GA などの直接探索法の利点の一つである．具体的にはペナルティの与え方としては以下のようなものがありうるだろう．

(1) 敵のぎりぎり近くをすり抜ける回数を抑制するため，“自分と同じ高さの 1 マス前に敵がいる”状態に対し，1 フレームあたり 1 点減点する．
(2) 足場ぎりぎりでジャンプする回数を抑制するため，ジャンプ時・着地時に，足場の端までの距離が 10 以下であれば，(10-距離)点減点する．
(3) 動作の切り替え回数を抑制するため，15 フレーム以内に動作が切り替わるたびに 1 点減点する．

これらはあくまで例であり，その閾値やペナルティの重さもチューニングされたものではない．将来的には，被験者実験で「どんなルートは模倣しにくいのか」を定量的に計測し，また各プレイヤーの技量や傾向に合わせてこれらをチューニングすべきだと考えている．

## 5. 遺伝的アルゴリズムによる最適化

前章では, ルートの表現方法, 評価方法について述べた．本章では，これらを用いて実際にルートを最適化するための手続きについて述べる．どのようなステージに対してルートを最適化するのかによって若干最適化手段も変わってくるが，本論文では比較的シンプルな「道中に 3 つの穴（致死断崖）を持つ，部分的な後戻りの必要がないステージ」を対象に採用している．このステージはベンチマークソフトがステージシード 0, ステージレベル 0 で生成するものに，3 つの穴を追加して作成した．図 5 にステージの一部を撮影し，連結したものを示す．狭い穴への滑り込みのようなテクニックは必要としないステージであり，ルートを構成する行動要素は表 1 から不要なものを削除して表 2 のように 6 つに減らした．ステージは 256 分割し，およそマス半分ごとに行動を切り替えることができるようにした．従って，ルートのとり得る範囲は 6^256 である．

図 5 本論文の対象となったステージの一部

最適化手法として採用するのは遺伝的アルゴリズムであり，世代交代モデルとしては初期収束の起こりにくい Minimal Generation Gap [7] を採用した．具体的な手続きはパラメータ込みで以下の通りである．

(1) 初期個体を 100 個体ランダムに生成・評価し，親世代とする．
(2) 親世代から 2 個体非復元抽出し，50 組のペアを作る．
(3) 各ペアから 8 個体の子個体を作成し，評価する．詳しい作成方法は次節で述べる．
(4) 親子 10 個体の中から, 評価値 1 位と 2 位を生存させる個体とする．
(5) 全てのペアで生存個体を設定したら, 次世代として(2)から（4）を繰り返す．
(6) 400 世代終了後，最も評価値の高い個体のルートを確認する．

表 2 採用した行動一覧

| | |
|---|---|
| 左 | 右 |
| 左＋ダッシュ | 右＋ダッシュ |
| 左＋ジャンプ＋ダッシュ | 右＋ジャンプ＋ダッシュ |

### 5.1 交叉と突然変異

交叉と突然変異は，新しい良い解を作るための遺伝的アルゴリズムの根幹操作である．交叉の場合は，両親の良い部分をいいとこどりした子個体を生成することを期待し，突然変異の場合は，（親が似通ってしまっているときに）悪い部分に良いものを導入することを期待している．この操作を不適切に定めると，良い親からも悪い子個体が頻繁に生成されたり，突然変異によって良い解が壊されたりする確率が高くなってしまう．

本論文ではそもそも，（時刻ごとではなく）X 座標ごとに行動を定めるということによって，交叉や突然変異が解の質を落とすことを防ごうとしている．時刻ごとの行動系列を持つのは表現としては自然に思えるが，1 か所でも行動を変えればそれが以降の全ての場所に影響を与えるという業を持つことになる．実際我々は予備実験として時刻ごとの行動系列による遺伝的アルゴリズムも実行しているが，クリアする解を見つけることすら困難であった．これを X 座標ごとにすることで，ある場所での変更が他の場所での挙動に大きな違いを与えないような工夫を行ったということである．

その上で，用いた交叉は一様交叉とした．すなわち，長さ 256 の配列 p1, p2 を親として，子の配列 c[i] は，p1[i] または p2[i]を i ごとにランダムに選ぶということである．突然変異については，3 つの方法を試みた．1 つめは，「一様に 10%の部位を変異させる」ものである．これだと解の破壊が頻発したため，2 つめとして「1 か所または 2 か所のみ変異させる」ものも試みた．さらに，クリアができない

解の改善を狙って，3 つめとして「マリオが敵にやられた，あるいは穴に落ちた場所の周囲 10 マスにだけ変異を起こす」ものも試みた．これらをどのように組み合わせたかについては 6 章で述べる．

## 6. 性能評価

5 章では，遺伝的アルゴリズムの枠組みと，交叉・突然変異操作について述べた．本章では，突然変異に工夫を行う場合と行わない場合でどのような差がでるか，さらに 4.4 節で述べたペナルティを加えるとどのような差がでるかについて比較実験を行う．比較を行う設定は以下の 4 つである．

A) ペナルティ・強制変異なし．変異発生時には全ての遺伝子がそれぞれ 10.0%の確率で変異する．

B) ペナルティなし．両親個体が未クリアであれば，死亡箇所の周辺 10 マスで強制的に変異を発生させる．それ以外の変異発生時には全ての遺伝子がそれぞれ 10.0%の確率で変異する．

C) ペナルティなし．両親個体が未クリアであれば，死亡箇所到達点の周辺 10 マスで強制的に変異を発生させる．変異発生時には 70.0%の確率でランダムな 1 カ所，30.0%の確率でランダムな連続した 2 カ所が変異する．

D) 自分と同じ高さの 1 マス前に敵がいる状態に対し，1 フレームあたり 1 点の減点．両親個体が未クリアであれば，死亡箇所の周辺 10 マスで強制的に変異を発生させる．変異発生時には 70.0%の確率でランダムな 1 カ所，30.0%の確率でランダムな連続した 2 カ所が変異する．

なお， A・B・C は（数式３），D では（数式１）と異なる評価関数を用いて最適化を行った．よって D については（数式１）によって得られた最良個体で再度テストプレイし，（数式３）で再評価して同一の基準に合わせたうえで比較を行った．

$$E = distance + 1024 + 10 \times leftTime \qquad (3)$$

これらの実験のうち，A と B は 20 試行の予備実験を行ったが，図 6 に示した実験 B の最良評価値推移のようなグラフの形状となった．最良値・平均値ともに，世代を重ねることによる性能向上がほぼ期待できなかったため，200 世代目で実験を打ち切っている．一方，C と D の実験では図 7（実験 C），図 8（実験 D）に示すように 400 世代目近くでも評価値が増加している．これは，10%の部分を変異させるという B の突然変異が破壊的であって殆ど良い解を作るのに寄与できないのに比べ，C では 1 または 2 か所に限定したために比較的高確率で改善解を得られているためだと予想できる．

また，A はクリアするルートの発生がかなり遅く，学習結果も不安定だったため，以後の実験では死亡箇所周辺に強制的な変異を加える手法へと変更した． そのため，本論文では B，C，D について比較する．

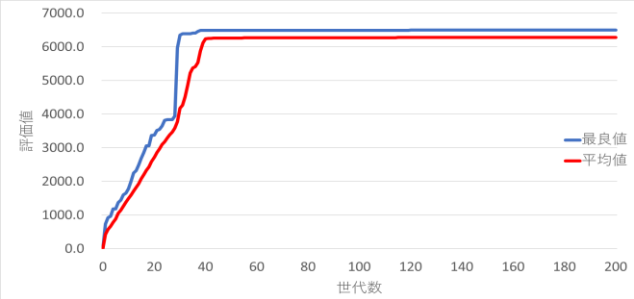


図 6 実験 B の最良評価値推移（200 世代）

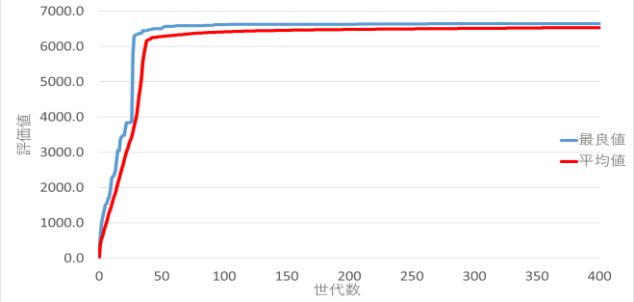


図 7 実験 C の最良評価値推移（400 世代）

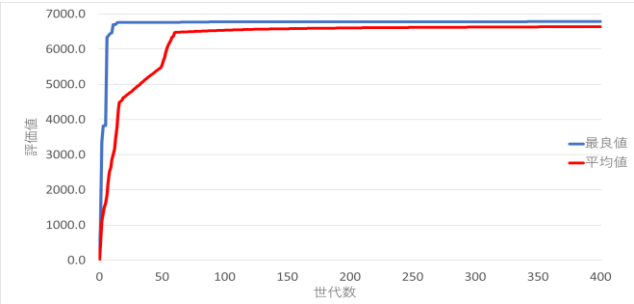


図 8 実験 D の最良評価値推移（400 世代）

B，C，D の手法はいずれもクリアするルートが安定的に生成されたため，全試行で最も高い評価値のルートについてその性能を検証し，ステージクリアまでのゲーム内時間である“クリアタイム”，敵と衝突した回数である“負傷数”，評価値，同じ高さの 1 マス前の敵とすれ違った（ニアミス）回数である“ペナルティ”を表 3 にまとめた．

表 3 各実験の最良ルート性能一覧

| | クリアタイム | 負傷数 | 評価値 | ペナルティ |
|---|---|---|---|---|
| 実験B | 61 | 2 | 6500 | 32 |
| 実験C | 46 | 2 | 6650 | 13 |
| 実験D | 32 | 2 | 6790 | 4 |

### 6.1 クリアタイム

3 実験のクリアタイムは意外なことに，行動に制約のない実験 B や実験 C ではなく，ペナルティによる制約のあるはずの実験 D から最も良い記録が得られた．各世代最良評価ルートのクリアタイム推移を全試行から取得し，図 9（実験 B），図 10（実験 C），図 11（実験 D）に示す．なお，スペースの都合上クリアタイム 110 以上のものは省略している．また，それぞれの図の最良値と平均値をまとめたものを図 12 に示す．

実験 D ルートの平均値は 200 世代目付近で実験 C の最良値とほぼ同タイムとなった．これはペナルティの導入で目の前に敵がいる状況が減り，通過するコースがずれても負傷しにくくなったためと考えられる．これにより交叉・変異の成功率が上がり，良い個体が誕生したと推測する．

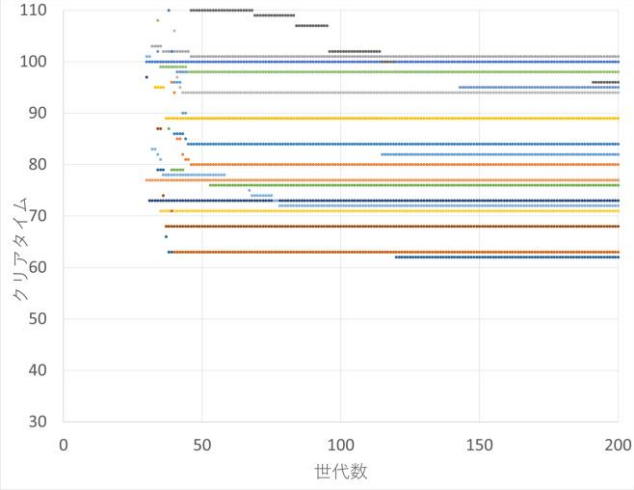


図 9　各世代最良評価ルートのクリアタイム推移（実験 B）

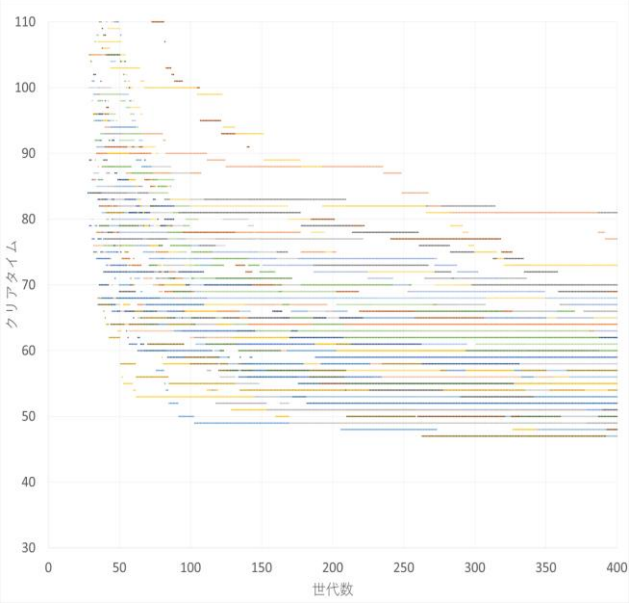


図 10　各世代最良評価ルートのクリアタイム推移(実験 C)

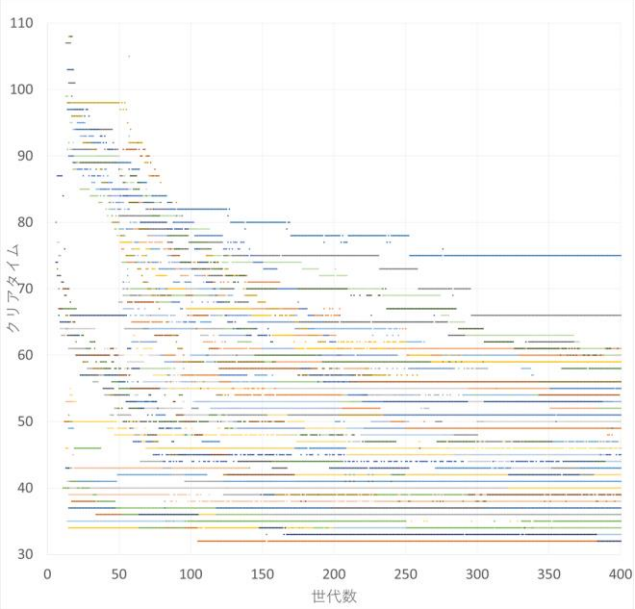


図 11　各世代最良評価ルートのクリアタイム推移(実験 D)

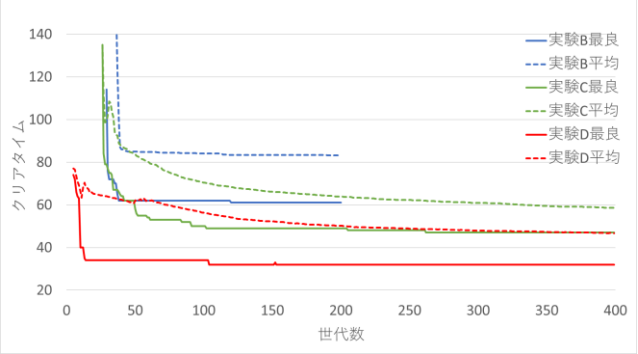


図 12　各実験における各世代最良評価ルートの
クリアタイムの最良値・平均値推移

### 6.2 ニアミス

各実験で得られたルートのプレイしやすさを検証するため，実験 B と実験 C でもテストプレイを行い，ペナルティが発生した回数を計測した．実験 D は評価関数にペナルティが含まれるため，3 実験の中では最も少ないが，実験 B から実験 C でも大幅に減少している． 実際にプレイしやすいルートかを検証するため, x 軸の各 256 座標における最も近い敵の距離を計測した．この距離は，マリオがアクションを取る際に，マリオの x,y 座標と敵の x,y 座標の差を取得してユークリッド距離を計測したものである．この際の座標は，ステージを x 軸方向に 4096 分割，y 軸方向に 256 分割した場合のものであるため，敵との距離をより詳細に得ることができる．この距離が 23 以下の場合には，マリオの周辺 1 マス以内のどこかに敵がいることとなる． 図 13 に距離のヒストグラムを示す．

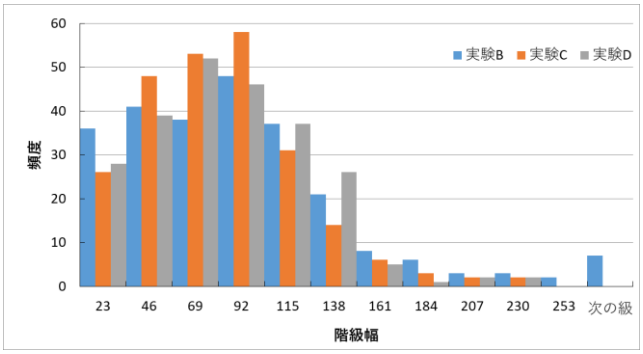


図 13　最も近い敵との距離ヒストグラム

実験 B のルートは，256 マス中 36 マスで敵が周囲 1 マス以内にいる状況だった．実験 C で 1 マス以内に敵がいた回数は 26 回だったのに対し実験 D では 28 回と，ペナルティの無い実験 C のほうが少ない結果となった．一方，同じ高さの 1 マス前に限れば表 3 で示した通り実験 D が一番少ない．これは，実験 D のルートが敵を踏みつけによって排除するため発生していると考えられる．図 13 の敵との距離は，踏みつけの直前にも計算されるためである．実験 C での踏みつけ回数は 12 回，実験 D では 16 回であり，これらの値を減算した場合には，実験 D の周囲一マスに敵がいた回数が最も少なくなる．

### 6.3 その他の観察

いずれの実験でもステージクリア時の状態はちびマリオであった．これは，いずれの実験でも評価値にマリオの状態を加えていないためであると考えられる．ただ，実験 B のマリオが立ち往生している間に負傷している一方，実験 C と D ではダッシュでステージを進む最中に負傷するといった違いがみられた．負傷してでもクリアタイム短縮を図る点はスピードランとしては非常に重要であり，この点から実験 C・実験 D は本研究の目的に沿った動きをしていると評価できる．

## 7. おわりに

本研究ではスピードランという，ユーザーによって“後から追加された目標”を達成する AI を，行動の最適化によって獲得できることが確認できた．今回は“素早いクリア”という目的に応じたスピードランルートの提案だけだが，評価値やペナルティを改良することで，“コインをすべて取る”や“負傷することなくクリア”などの目標にも対応できると考えられる．

今後の展望として，個人にとって最適なルートやスピードラン特有のテクニックを用いるルートの探索を行いたい．本研究の手法は，交叉や突然変異手法やペナルティの与え方などでの改良の余地は大きいため，今後はこの点について研究したい．

## 参考文献

[1] Julian Togelius, Sergey Karakovskiy and Robin Baumgarten, The 2009 Mario AI Competition, Evolutionary Computation pp.1-8, 2010-07

[2] David Silver et al., Mastering the Game of Go without Human Knowledge, Nature 550, pp.354-359, 2017-10

[3] 藤井叙人，佐藤祐一，中嶌洋輔，若間弘典，風井浩志，片寄 晴弘，生物学的制約の導入による「人間らしい」振る舞いを伴うゲーム AI の自律獲得, 第 18 回 GPW, pp.73-80, 2013-11

[4] Mario AI Championship, Mario AI Benchmark revision 762 v.14, 入手先< https://sites.google.com/a/marioai.com/www/marioai-benchmark/download >（アクセス 2019-10-8）

[5] Games Done Quick, https://gamesdonequick.com（アクセス 2019-10-8）

[6] Julian Togelius, Sergey Karakovskiy, Jan Koutnik, Jurgen Schmidhuber, Super Mario Evolution, 2009 IEEE Symposium on Computational Intelligence and Games, pp.156-161, 2009-09

[7] 佐藤浩，小野功，小林重信，遺伝的アルゴリズムにおける世代交代モデルの提案と評価, 人工知能学会誌 Vol.12 No.5, pp.734-744, 1997-09