Vol.2020-GI-43 No.7
2020/3/13

RPGのマップ探索におけるエンカウントを考慮した最短時間導出自動化スクリプトの作成

三明 主佳[†1]　橋本 剛[†1]

概要：デジタルゲームでは，複雑化に伴い，不具合修正やゲームバランス調整などの品質保証にかかる負担が増大している．この品質保証は通常，手動で行われていることが多く，自動化の試みはまだ少ない．GAを用いてゲームバランスを確認する手法が提案されているが，ゲームをやりこんだプレイヤーがどのような動きをするかを予測することは難しい．一方，TASと呼ばれる，ツールを使ってできるだけ速いゲームクリアを目標としたプレイヤーが増えているが，理想の結果を得るために膨大な手作業を必要としている．本稿では，このような作業を自動化させるために，RPGのマップ探索に焦点を当て，エンカウントを考慮したIDA*を用いて最短時間を導出する手法を提案する．また，提案手法を実装したスクリプトを用いて実験を行い，その実用性を論じる．

An Automatic Shortest Time Deriver Script for Search on RPG Maps Considering Encounters

Kazuyoshi Miake[†1]　Tsuyoshi Hashimoto[†1]

1. はじめに

近年，国内のゲーム市場は増えつつあり，各社の競争は激しくなっている．それに伴い，ゲームという商品に対して，以前の作品や別会社のゲームよりも高品質なものを作ることが求められる傾向にある．ゲームの開発には様々な場面でゲームに搭載した機能や仕様に対してのテストプレイが必要不可欠になっている．このテストプレイには当然人員や時間を割くため，開発の効率化に向けてゲームのテストプレイを自動化させることが求められる．ゲームのテストプレイを行うような研究はいくつか行われており，既存ゲームに実装されているキャラクターの性能比較を行うために疑似プレイヤーによる戦闘データの統計を取る研究 [1] や 3D のフィールド上をプレイヤーキャラクターが移動し，設置したオブジェクトをすり抜けてしまうバグが無いかを確認する研究 [2] などがある．一方で，ゲームのクリアタイムを競う「タイムアタック」を行うプレイヤーが増えつつある．このようなタイムアタックプレイヤーもまた，ゲームに対して何度もテストプレイを繰り返し，どの場面で，あるいはどうやってクリアタイムの短縮ができるかを試行錯誤している．ゲームの開発側にとっても，タイムアタックプレイヤーが欲しているようなゲームの最短時間を指標として必要とされていることがある．しかし，ゲーム開発の効率化を求めている研究で最短時間を導出することを重要視しているものは少ない．

そこで本研究では，ゲーム開発側やタイムアタックプレイヤー側のどちらにも必要とされることの多い，ゲームの各場面における最短時間の導出を自動で行わせることを目的とする．

なお，最短時間を導出する対象は，ゲームの中でもメジャーなジャンルで最短時間を要することの多いマップ移動が頻繁に現れるロールプレイングゲーム (RPG) とし，実際にゲーム上で見られるような特徴を持ったマップ上の移動に要する最短時間の導出を行う．この最短時間の導出は最短経路探索 [3] や 15 パズル問題 [4] に用いられることの多いアルゴリズムである IDA*を扱う．

[†1] 現在，松江工業高等専門学校
Presently with National Institute of Technology, Matsue College

表 1　勝利編成に含まれるキャラクターの利用頻度
Table 1　Usage frequency of characters in the winning formation

順位	頻度	利用数	平均戦闘時間 (秒)
1	0.71	138407	533.47
2	0.58	114426	536.66
3	0.45	87945	498.04
4	0.32	62871	517.22
5	0.32	61894	577.96

2. 関連研究

既存スマホゲームのゲームバランス調査の自動化を目的とする研究 [1] では，遺伝的アルゴリズム (GA) を用いて様々な戦闘用の編成を作り，特定の敵との戦闘を行わせていった．その後，勝利編成に含まれるキャラクターや装備の統計をとり，勝利編成や戦闘時間の短い編成に頻出しているキャラクターを抽出した (**表 1**)．勝利編成に 7 割近く利用されているキャラクターもいれば，採用率が極端に高くないが，戦闘時間が短くなる編成に含まれることの多いキャラクターも抽出された．この研究では全体的な傾向を見ていくものであったため，GA で各個体を生成しても，平均的な戦闘時間を重要視して統計を取っていた．しかし，どの世代まで繰り返すかにもよるが，GA で作られていった個体による戦闘時間がその戦闘の最短時間を記録しているとは限らない．例えば，あまり使われていないキャラクターのみで組み合わせた編成が偶然にも速く戦闘勝利できることがあるかもしれない．その場合，そのキャラクターごとの役割に対応するキャラクターの追加が今後のゲームバランスを左右される可能性もある．このように，ゲームのテストプレイには平均的なデータも必要だが，最短時間のような極端な数値も指標として必要となる．

3. RPG のエンカウントについて

3.1 RPG とは

ロールプレイングゲーム (RPG) は，プレイヤーがゲーム内の架空世界上で繰り広げられるストーリーを，ゲーム専用のキャラクターを操作して進めていくことが主体となるゲームジャンルの 1 つである．現在までに「ポケットモンスター[*1]」や「ドラゴンクエスト[*2]」,「ファイナルファンタジー[*3]」など，名のあるゲームシリーズが登場している．

3.2 エンカウントとは

RPG では，プレイヤーが操作するキャラクターがストーリー上の難題を攻略するために育成が必要とされることが多い．この育成の多くは敵キャラクターとの戦闘に勝利して得られる「経験値」を積み重ねていくものである．また，この敵キャラクターとの戦闘の発生を「エンカウント」と呼び，これは主に「ランダムエンカウント」と「シンボルエンカウント」に分けられる．

まず，ゲーム内でプレイヤー自身が敵キャラクターを視認できる形式のものを「シンボルエンカウント」と呼び，基本的にどのタイミングでどのような敵とのエンカウントが発生するかを想定しやすい傾向がある．一方で，敵キャラクターを視認できないエンカウントを「ランダムエンカウント」と呼ぶ．これは，ゲームのマップ上を移動している際に，予め用意された疑似乱数などによってどの敵と，どのタイミングでエンカウントするかが決められるものである．そのため，狙った敵とのエンカウントやエンカウントの回避をすることが困難である．このエンカウント判定を行うための疑似乱数などの設定は比較的重要視されることが多く，エンカウントする確率が高すぎると「目的地になかなか辿り着けない」「戦闘から逃げることが面倒」，低すぎると「育成が捗らない」「目的の敵と出会えない」といったプレイヤーの不満点を生じないようなエンカウント率が求められる．ゲーム開発側は，この設定が適切かどうかを調べる際には人力でプレイしていく必要がある．また，タイムアタックプレイヤーにとっては不要な敵とのエンカウントはできるだけ避けることが必要となる．本研究では，ゲーム内のプレイヤーにとって基本的に予測のできないランダムエンカウントを絡めたマップ移動における最短経路，最短時間を導出し，マップ移動を行うテストプレイの自動化に扱えるような指標を得る．

4. IDA*

「IDA*」は深さ優先探索に探索する範囲の制限や反復深化，ヒューリスティック要素を加えた探索アルゴリズムである．このアルゴリズムは探索する際に，開始地点から終了地点までに着いた各地点でコスト関数 $f(n)$，$g(n)$，$h(n)$ を用いた以下の式による評価値を算出する．[5]

$$f(n) = g(n) + h(n)$$

$f(n)$: 地点 n の総評価値
$g(n)$: 開始地点から地点 n までに要するコスト
$h(n)$: 地点 n から終了地点までに要する推定コスト

これらのコスト関数を用いた IDA*の探索は以下のようにして実施される．

(1) 開始地点から終了地点までの $f(0)$ を算出し，探索する深さの制限を定める
(2) 現地点から移動できる地点を抽出する
(3) 2 で見つかった各地点の $f(n)$ を算出する
(4) $f(0) < f(n)$ ならその経路以降の探索を中止し，$f(0)$

[*1] Nintendo/Creatures Inc./GAME FREAK Inc. ポケットモンスターシリーズ,1995-2020.Video Game
[*2] SQUARE ENIX Co.,Ltd. ドラゴンクエストシリーズ,1986-2020.Video Game
[*3] SQUARE ENIX Co.,Ltd. ファイナルファンタジーシリーズ,1987-2020.Video Game

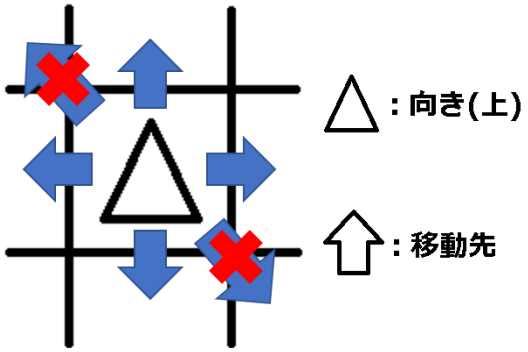

図 1　1 歩進むプレイヤーの移動先
Fig. 1　Movement range of player who moves one step

≧ $f(n)$ ならその経路以降の探索を継続する
(5) 制限した深さになるまで探索範囲を広げる
(6) 終了地点が見つからなかった場合は，見つかるまで深さを 1 増やして 2〜5 の処理を繰り返す

5. 問題点

5.1 一般的な IDA*の場合

IDA*は最短経路探索等で用いられているため，探索対象の距離を評価値としていることが多い．しかし，本研究では最短経路だけでなく最短時間の導出を目的としているため，距離だけを評価値とした場合は最短時間の導出は困難である．RPG を例に挙げると，前述したエンカウントのような経路の途中で生じるゲーム内のイベントによって，距離としては最短であっても，実際に経路を進む時間が長くなる場合がある．

5.2 対策

本研究ではこの問題への対策として，「距離」のみを評価値とする IDA*ではなく，「ゲーム内時間」を評価値とする．これによりエンカウントのようなゲーム内の時間がある程度進むイベントが発生する経路を最短経路から除外することができる．

6. 探索対象の仕様

本研究ではポケットモンスターシリーズのようなマップを経路探索の対象とし，RPG の仕様は以下のように仮定して行う．

- プレイヤーは上下左右の向きを持ち，その方向にのみ進む
  左上や右下のような斜め方向への移動は 1 回では行えず，左，上のように 2 回に分けて移動する (**図 1**)
- 歩行に必要なフレーム数は 1 歩目の歩き始めと，2 歩目以降の歩行の継続を区別するために以下のようにする
  1 歩目：同じ向きに進む場合は 14 フレーム，異なる向きに進む場合は 20 フレーム
  2 歩目以降：向きを問わず 16 フレーム例) 左を向いて静止している状態で
  左，左，下と進む場合は $14+2*16$ の 46 フレーム
  下，左，左と進む場合は $20+2*16$ の 52 フレームを要する (**図 2**)
- 移動マップは特定のエリア内でランダムエンカウントが発生し，シンボルエンカウントは発生しない
- 各地点のエンカウントの判定に 20 フレーム要する
- エンカウント判定は歩き始めた際に適当なフレームを定め，そのフレームが経過した際にエンカウントエリアに位置した場合にエンカウントする (**図 3**)

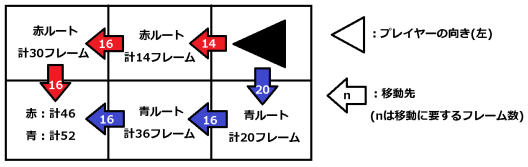

図 2　向きによって異なる総フレーム数
Fig. 2　Total number of frames according to direction

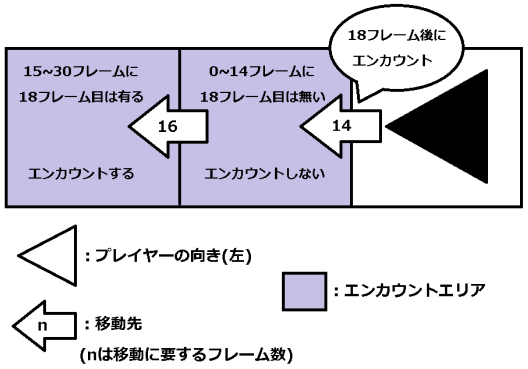

図 3　エンカウント判定の例
Fig. 3　Example of encounter determination

7. 実験用の IDA*

ゲーム内の時間を考慮して実装した IDA*は以下のようにした．(図 4)

- 距離はマンハッタン距離を扱う
- コスト関数 $g(n)$ は地点 n に辿り着くまでの総フレーム数とする
- コスト関数 $h(n)$ は (ゴールまでの距離)*(1 歩進むのに必要なフレーム数) とする
- 深さはプレイヤーの総移動フレーム数とする
- 深さの初期値は初期地点の $h(n)$ とし，反復深化させる際にはこの値を 1 フレームずつ増やす

8. 実験

本研究では，RPG に見られるようなマップにおけるス

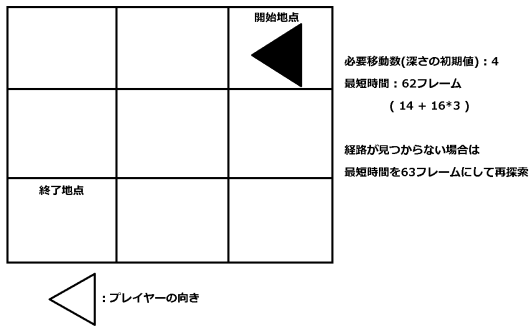

図 4　実装した IDA*の例
Fig. 4　Example of implemented IDA *

図 5　ゲーム上の探索範囲
Fig. 5　Search range on the game

タート地点からゴール地点までの最短経路，最短時間を導出する．前述したようにポケットモンスターシリーズに似たマップを対象とするために『ポケットモンスター ソウルシルバー[*4]』を，実行環境として DeSmuME[6] を用い，IDA*を実装したスクリプトは Lua で記述した．

8.1　探索マップ 1

まず，ゲームの移動に必ずしもエンカウントのような不意に時間を取られるようなイベントが発生するとは限らないため，**図 5** のようなマップ上の最短経路，最短時間を導出する．なお，探索範囲として図 5 の緑枠内を座標化した**図 6** を用意し，開始地点は青枠の，終了地点は赤枠の地点とする．

8.2　探索マップ 2

次に，本研究で最短時間を導出する要因となったエンカウントする範囲を持つ**図 7** のようなマップ上の最短経路，最短時間を導出する．なお，探索範囲として図 7 の緑枠内を座標化した**図 8** を用意し，開始地点は青枠の，終了地点は赤枠の地点とする．なお，黄枠ではランダムエンカウントが発生し，黒四角部分は移動できない地点とする．

[*4] 『ポケットモンスター ソウルシルバー』, ゲームフリーク, 任天堂, 株式会社ポケモン, 2009. (ニンテンドー DS)

図 6　座標上の探索範囲
Fig. 6　Search range on coordinates

図 7　ゲーム上の探索範囲
Fig. 7　Search range on the game

図 8　座標上の探索範囲
Fig. 8　Search range on coordinates

8.3　探索マップ 3

最後に，最短経路を導出するのに迂回を必要とする**図 9** のようなマップ上の最短経路，最短時間を導出する．なお，探索範囲として図 9 の緑枠内を座標化した**図 10** を用意し，開始地点は青枠の，終了地点は赤枠の地点とする．なお，探索マップ 2 と同様に黒四角部分は移動できない地点とする．

9.　結果と考察

9.1　探索マップ 1

探索結果として導出された経路は**図 11** の通りである．この結果から，同じ向きに歩き始めた経路が最短経路となっ

図 9　ゲーム上の探索範囲
Fig. 9　Search range on the game

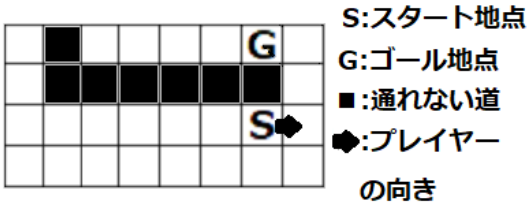

図 10　座標上の探索範囲
Fig. 10　Search range on coordinates

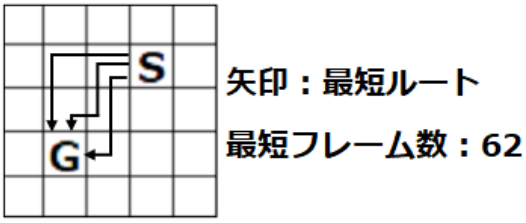

図 11　探索結果
Fig. 11　Result of search

ていることが見て取られる．これは，今回実装したゲームの仕様である「歩き始める方向によって必要なフレーム数が異なること」に左右されたためである．したがって，今回仮定した仕様上での最短経路は，同じ方向に向けて歩き始める経路が主体になると考えられる．

9.2　探索マップ 2

探索結果として導出された経路は図 **12** の通りである．この結果から，最短経路となった部分は同じ方向に歩き始め，エンカウントエリアから脱してゴールに向かうもののみであることが見て取られる．これは，歩き始めた際に定められた「エンカウントするフレーム数」が 5〜6 歩目の間となったため，5 歩以内にエンカウントエリアを脱したものはエンカウントせず，6 歩以降もエンカウントエリア内に位置していたものがエンカウントしたためである．例えば，左に 3 歩進んで図の赤丸のエリア内に位置したとする．その後ゴールへ進むには最短経路を進んだとしてもエンカウントエリアを 3 歩進むことになる．こうして 6 歩目

図 12　探索結果
Fig. 12　Result of search

図 13　探索結果
Fig. 13　Result of search

表 2　探索時間
Table 2　Time of search

探索マップ	実行時間 (秒)
1	141
2	673
3	964

にエンカウントエリアに位置するためエンカウントが発生し．最短経路から外れることになる．

また，実験 1 で示したように，左を向いているプレイヤーが下方向に移動しようとした際には少し長いフレーム数を要するため，1 歩目に下へ進む経路は最短経路から除外される．

9.3　探索マップ 3

探索結果として導出された経路は図 **13** の通りである．この結果から，障害物に阻まれるマップも迂回して最短経路を導出できることが見て取られる．このマップは経路としては 4 歩移動するものであったのに対して，同じ 4 歩進む実験 1 や明らかに 4 歩以上必要となる実験 2 のマップよりも探索時間が長くなった．これは最短時間に定義した値が「障害物を考慮しないマンハッタン距離」を扱ったため，反復深化する回数が過剰に必要となったことが原因と考えられる．

9.4　探索時間

探索マップ 1〜3 の探索に要した実行時間 (少数以下切り上げ) を表 **2** に示す．

10.　まとめ

本研究では RPG のマップ移動を対象にして「ゲーム開発中に設定された値の調整やタイムアタックの最短経路を通る最短時間の導出の自動化」を目標として，IDA*を用いた最短経路かつ最短時間の移動経路を導出するスクリプ

トの作成を行った．結果として RPG におけるマップ移動の最短時間を導出できた．本研究では RPG のみを対象としたが，探索部分のパラメータや関数の改変を行えば別のゲームジャンルのテストプレイに転用することが可能であると考えられる．

11. 課題と展望

11.1 課題

実験 3 の結果から，障害物を考慮した移動経路の導出に必要な探索時間が膨大となることが課題の 1 つとなる．実験 3 の場合はゴールまでのマンハッタン距離は 2 であるが，実際には最低でも 4 歩の移動が必要となる．この対策としては，障害物を含めて最低でもどのくらいの歩数が必要か，「距離」を評価値とした最短経路を出すことで導出し，その歩数を探索のパラメータに導入した「ゲーム内時間」を評価値とした最短時間の経路探索を行うことが対策例となる．本研究内では間に合わなかったが，学会発表の際にはこの対策案を実装し，同様のマップ探索を行ったことで探索時間がどのように変わったかを紹介する．

また，本研究ではエンカウントのみを考慮したが，実際の RPG では最短経路を進む際にエンカウント以外のイベントが挟まれることがあるため，各イベント用に探索部分を改変したり，小規模の範囲を主な用途として扱ったりすることが対策と考えられる．

11.2 展望

本研究で作成したスクリプトでは，探索時間やイベント別の対策がどのくらい必要なのか予想のしづらいことへの課題があるが，基本的なマップへの探索は可能である．現在実装した段階ではマップの範囲が小区間の探索には対応できる．これにより，タイムアタックプレイヤーが小区間のタイム更新を行う際に何度も手動でやり直す手間を省くことができる．また，敵とのエンカウントで最短経路から外れることから，結果のフレーム数が想定したものより大きくなれば，エンカウント率が高く，序盤で設定した最短時間と近い結果が出れば，エンカウント無しで目的地に辿り着くことができるといったエンカウント率の確認テストを行うことができる．これらはまだ小区間内の対象に限られるが将来的には様々なテストプレイを自動化させることに繋がっていくと考えられる．また，探索に必要なパラメータ等を直接取り扱っている開発者側であれば，より適した探索スクリプトに改変することは可能であり，最短時間が欲しいタイミングで発生し得るイベントが既知であるタイムアタックプレイヤー側は有限個のイベントにも対応できる．どちら側の立場においても様々なゲームに対しての利用が可能であると考えられる．

参考文献

[1] 眞鍋和子, 三宅陽一郎: 遺伝的アルゴリズムを用いたプレイヤーエージェントによるデジタルゲームのバランス調整, SIG-SAI, 30(2), pp. 1-6, 2017
[2] 茂原敦之, 水口充: 汎用ゲームエンジンに外付けするデバッグ環境の提案, EC2018 論文集, 2018, pp. 31-35, 2018
[3] 早川広記: IDA*アルゴリズム（Iterative Deepening A* algorithm）の計算リソース多種混在ハードウェア向けの最適な実装方法の検討, 電気通信大学修士論文, 2016
[4] 山本修身, 佐藤根寛: ギャップ集合を用いた 15 パズルの最適解探索の高速化, 人工知能学会論文誌, 26(2), pp. 419-426, 2011
[5] Korf, R. E.: Depth-first iterative-deepening: an optimal admissible tree search, Artificial Intelligence, 27(1), pp. 97-109, 1985
[6] DeSmuME, http://desmume.org/