2021 年度　卒　業　論　文

スタート地点とゴール地点が変化した際の経路探索手法に関する研究

指導教員：渡辺 大地 教授

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0118279
吉田 涼真

2021 年 7 月

2021 年度　　卒　業　論　文　概　要

論文題目

スタート地点とゴール地点が変化した際の経路探索手法に関する研究

メディア学部 **学籍番号：M0118279**	**氏名**	吉田 涼真	**指導教員**	渡辺 大地 教授

キーワード	人工知能、経路探索、コストマップ、 再探索、アルゴリズム

経路探索は、ある地点から目的地点への移動経路を算出するものである。使用用途は、ゲーム AI やロボット AI、カーナビゲーションシステムなどがある。主な手法として、ダイクストラ法や A*アルゴリズムなどがある。この 2 つの手法では、ゴール地点が最短経路から外に移動した際に経路探索を再び行う必要がある。Moving Target D* Lite という手法では、そのような状態で、マップに変化が生じた際に、最短経路を高速に発見することができる探索手法である。しかし、この手法は、スタート地点が最短経路から外れてしまうと成り立たなくなってしまう。ゲームやロボットの分野では、スタート地点やゴール地点が最短経路から外れて移動することが頻繁に起こるため、何度も全探索を行う必要がある。本研究は、スタート地点やゴール地点が最短経路から外れた際に先行手法よりも速く最短経路を算出できるような探索手法を提案することを目的とする。

提案手法では、ダイクストラ法による探索情報を事前データとして記録しておき、スタート地点からゴール地点へのベクトルに合わせて事前データを採用する。事前データをもとにコストの更新が必要なマスに対しては、追加移動コストとしてコストを保存し、探索を行う。更新不要なマスに関しては、処理を減らして探索を行うことができるため、探索の高速が可能となる。本手法の検証を行った結果、山道のようにほぼ一本道やビル街のように規則的に障害物があるマップに対して新しいスタート地点が前の最短経路外に移動する場合、有用であることが判明した。

目 次

図 目 次

第 1 章

はじめに

1.1 背景と目的

経路探索とは、ある地点から目的地点への移動経路を算出するものである。使用用途は多岐にわたる。主な使用例として、ゲーム AI[1][2]、ロボット AI[3][4]、カーナビゲーションシステム [5] や電車乗り換え案内アプリケーション [6]、災害時における経路シミュレーション [7] などに用いられる。

ゲーム AI の分野において、池田ら [8] は、アクションゲームにおける、ある特定のプレイヤの行動の特徴を分析し、その特徴を模倣する AI プレイヤの作成方法について研究した。この手法は、事例ベース政策最適化を用い、特徴を強調することで、特定プレイヤの摸倣対象の行動を経路探索によって導くことができるものである。また、佐藤ら [9] は、Influence Map を用いて、シューティングゲームにおける危険な場所を特定し、人間らしく弾避けを行う AI プレイヤの作成について提案した。

ロボット AI の分野では、中宮ら [10] は、移動型センサノードを使用して経路探索を行う際に起こる問題をコストマップを用いることで、解決を図る研究を行った。

カーナビゲーションシステムの分野では、杉山ら [11] は、救急車が患者を乗せた後に病院までどのような経路で向かうかをデータベースを利用することで、高速な経路探索を実現した。この手法は、目的地点が病院と決まっているため、病院までの主要になる道路を通る経路を事前にデータベースに登録しておくことで、渋滞回避などの再探索にも高速に対処できるという特徴がある。

カーナビゲーションシステムやロボット AI など様々な分野で用いられている手法の一つに、ダ

イクストラ法 [12][13] がある。ダイクストラ法とは、1959 年にエドガー・ダイクストラにより考案され、ノード間におけるコストをもとに 2 点間の最適な経路を算出するものである。乗り換え案内システム [14] もこの手法を使用している。この手法を用いることで、駅間における交通費や所要時間をコストとして、それぞれの最適な経路を探索し、費用優先や時間優先などのユーザーが自分好みの経路を選択できるようにしている。図 1.1 は、ダイクストラ法の一般的な模式図を示す。

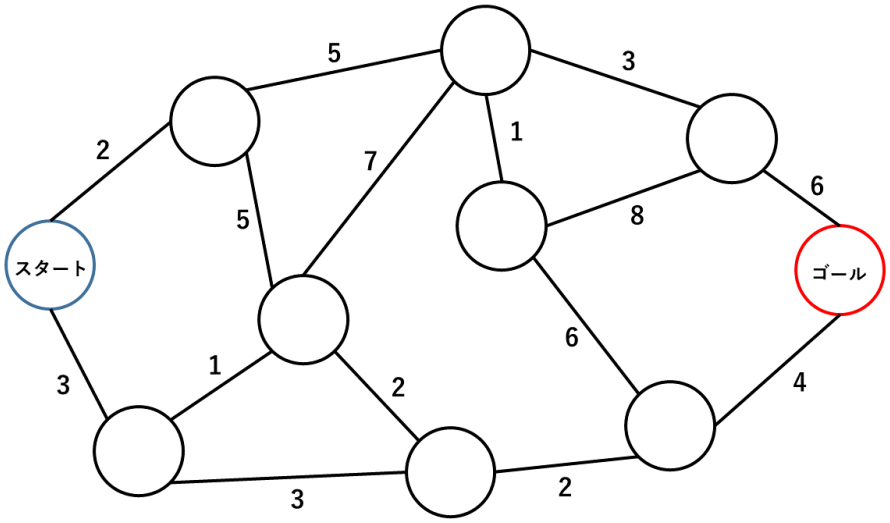

図 1.1　ダイクストラ法の模式図

円はノードと呼ばれる地点で、それらのノード間を結ぶ直線がエッジという道を示しており、エッジに付随している数値が、コストである。

この手法とは別に、ゲーム AI や他の多くの分野にも用いられる手法として、A*アルゴリズム [15] がある。A*アルゴリズムはダイクストラ法の改良版のアルゴリズムである。

これまでのカーナビゲーションシステムや電車乗り換え案内システムでは、探索の開始地点となるスタート地点と、目的地点を表すゴール地点の変更が頻繁には行われることはなかった。しかし、ゲーム AI やロボット AI の分野では、スタート地点やゴール地点が頻繁に更新され、そのたびに経路探索が必要とされる場面がある。両方の地点が別な地点へと遷移すると、スタート地

点から各地点までにかかるコストを算出したものが使用できなくなる。そのため、スタート地点とゴール地点が遷移するたびに再探索を行う必要があるが、そのたびにマップ全体に対して再探索を行うことは処理時間が大幅にかかり、非効率なものとなる。

本研究の目的は、スタート地点とゴール地点がそれぞれ別なノードへ遷移した際に、事前の探索データを用いることで、一から探索するよりも速く最短経路を算出できる手法についての提案を行うことである。

1.2 論文構成

本論文は、全 5 章にて構成する。構成は 2 章にて、探索アルゴリズムについて述べ、3 章では、提案手法について述べる。また、4 章にて評価と分析について述べ、そして 5 章にてまとめを述べる。

第 2 章

経路探索の先行研究

本章では、経路探索の先行手法について説明する。2.1 節では、マップについての説明をして、2.2 節では、探索アルゴリズムについての説明を行い、2.3 節では、現状の問題点について説明する。

2.1　マップについて

本研究では、マップを格子状のマスで区切ったもので表現を行う。マップは、経路として移動可能なマスを通常マス、移動不可能なマスを壁マスとし、スタート地点を表すスタートマス、ゴール地点を表すゴールマスの 4 つから構成されている。図 2.1 は、格子状に区切られたマップを示している。本章における、以下に示す図 2.1〜2.12 は、経路探索アルゴリズムを格子状のマスで簡易的に表現したものである。

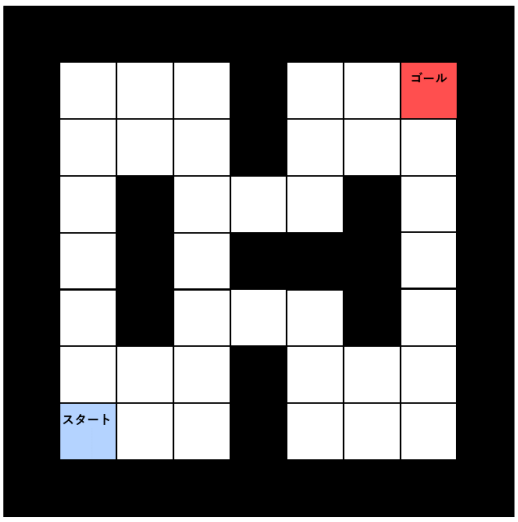

図 2.1　格子状のマップ

マップを構成する 4 つのマスはそれぞれ、通常マスが白色、壁マスが黒色、スタートマスが青色、ゴールマスが赤色で示してある。また、このマップでは、斜め移動や 2 マス以上の移動は考慮せず、上下左右の方向のマスにのみ進行可能である。

2.2 探索アルゴリズム

経路探索とは、スタート地点からゴール地点までの経路を、各マスのコストを計算することによって求めるものである。一般的に、経路探索では、スタート地点からゴール地点までの最小コストを求めることで、最短経路を算出している。他にも、コストの基準を変えることで、所要時間をより短くする経路や、移動にかかる費用をより少なくする経路を導き出すこともできる。経路探索では、最適な経路を計算するコストによって決定している。本節では、スタート地点からゴール地点までの最短経路を求める際に、用いられる手法について紹介する。

2.2.1 ダイクストラ法

ダイクストラ法では、スタート地点の周りに存在する通常マスに、数値を割り振っていく。その際の、数値はスタート地点からそのマスまでの移動コストを表しており、一度数値が割り振られたマスに関しては、移動コストが上書きされることはない。この移動コストとは、スタートマスからあるマスへ移動する際にかかるコストのことを示す。以後、割り振られた数値のことをコストと呼ぶ。まず、スタート地点の周りにある通常マスに数値を入れる。このとき、スタート地点は先にコスト 0 が割り振られる。その様子が図 2.2 である。

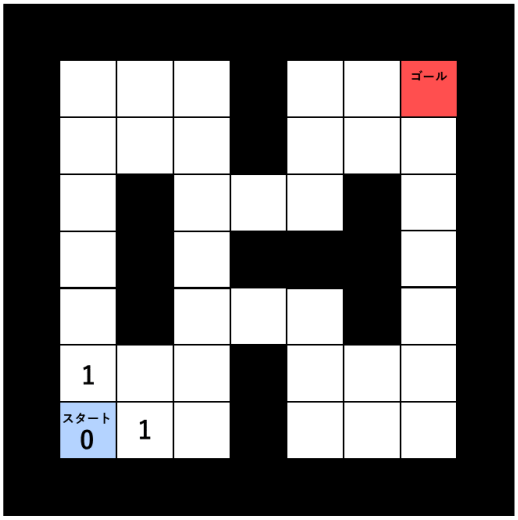

図 2.2　探索の初期段階

今度は、コスト 1 を割り振られたマスの上下左右のマスに対して同じ作業を行っていく。この作業をすべてのマスに対して行っていき、すべての通常マスにコストが割り振られた時点で処理を終了する。コストの割り振りが終わった様子を図 2.3 に示す。

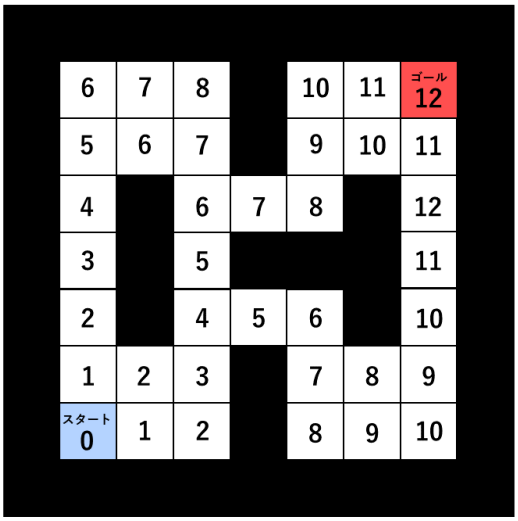

図 2.3　探索準備完了

そして、ゴール地点からコストが低くなるようにマスを遡っていき、スタート地点にたどり着くまで現在のマスよりもコストの小さくなるマスへ遡る。遡る際に、同じコストのマスが表れる

ことがあるが、その場合はどちらのマスを選択しても問題なく処理が行える。最終的にスタート地点までに経由したマスが最短経路となる。最短経路が算出された様子を図 2.4 に示す。最短経路となるマスはピンク色で示す。

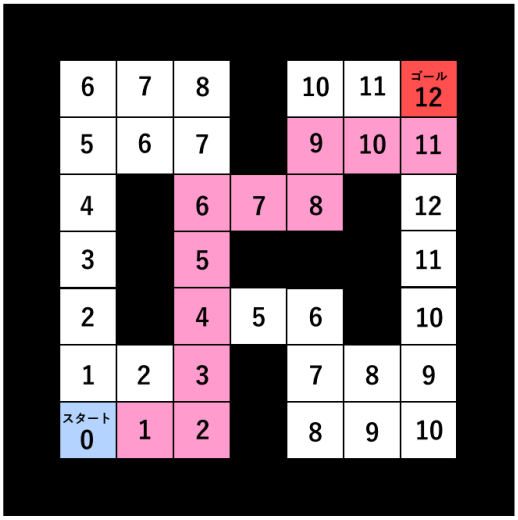

図 2.4　最短経路の算出結果

ダイクストラ法では、このようにマップ全体に対して、探索を行うことで最短経路の算出を行う。また、コストの種類を変えることでそれぞれの最適経路の算出を行う。スタート地点が変わらない限り、このコストマップは再利用することが可能である。そのため、ゴール位置だけが計算済みのマスに移動した際は再探索が容易にできる。新たなゴール地点からスタート地点へコストが小さくなるようにマスを遡るだけで、最短経路を算出することができるからである。ダイクストラ法によって算出される経路は正確だが、マップ全体に探索を行うため、処理時間が大幅にかかってしまうという問題点がある。

2.2.2　A*アルゴリズム

A*アルゴリズムは、第 2.2.1 項にて述べたダイクストラ法の改良版となるアルゴリズムである。この手法は、スタート地点からゴール地点までコストを計算するという基本的な処理の流れはダイクストラ法と同じである。しかし、探索をする際にダイクストラ法では行わなかった処理を同時

に行っている。A*アルゴリズムは、ヒューリスティックコスト [16] を使って探索を行う。ヒューリスティックコストは、ある程度のレベルで正解に近い解を得ることができる推定値のことである。ヒューリスティックコストを利用することで、ダイクストラ法より早く最短経路を見つけることができる。ヒューリスティックコストの計算方法は様々で、マンハッタン距離やユークリッド距離などを使うことができる。格子状のコストマップで一般的に使われるのが、マンハッタン距離である。マンハッタン距離によるヒューリスティックの値を h、現在探索中のマスの座標を (x, y) とし、ゴール地点の座標を (X, Y) とすると、h は式 (2.1) より求めることができる。

$$h = |X - x| + |Y - y|. \tag{2.1}$$

マンハッタン距離は、現在探索中のマスの座標とゴール地点の座標をもとに算出するものであるため、ゴール地点の位置が変化すると各マスにおけるヒューリスティックコストは変わる。

ここで、実際に図 2.1 におけるスタート地点の座標を $(0, 0)$ としたとき、ヒューリスティックコストがどのようになるのかを示したものが、図 2.5 である。ただし、スタート地点の右側を x 座標の正方向、上側を y 座標の正方向とする。

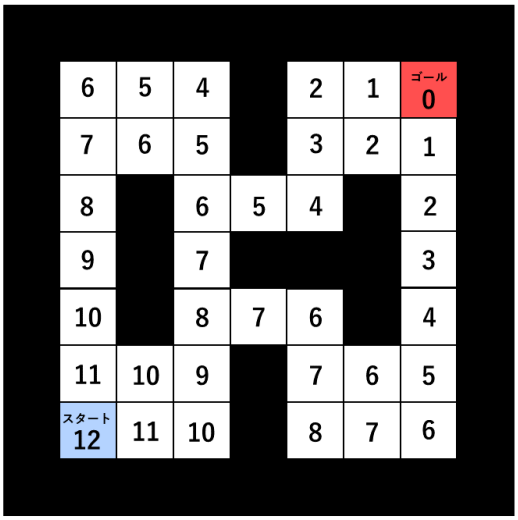

図 2.5　ヒューリスティックコスト

スタート地点のヒューリスティックコストは、$h = |6 - 0| + |6 - 0|$ より $h = 12$ となる。ゴー

ル地点のヒューリスティックコストは、$h = |0-0| + |0-0|$ より $h = 0$ となる。A*アルゴリズムは、探索する際にスタート地点からヒューリスティックコストとダイクストラ法のコストを求め、各マスの合計コストが低くなるマスから計算をする。そのため、実際には図 2.5 のように、最初から全体のヒューリスティックコストは計算されていない。その途中段階の様子を示したものが図 2.6 である。左の数値がダイクストラ法によるコストを表しており、右の数値がヒューリスティックコストを示している。

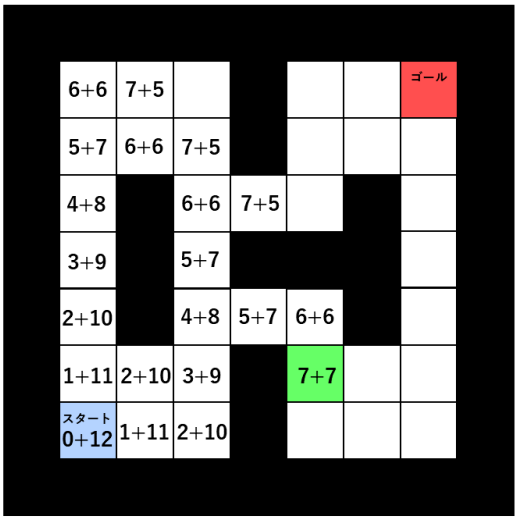

図 2.6　A*アルゴリズムの探索の途中段階

図 2.6 は、ダイクストラ法のコストが 0〜7 までのマスに対して探索を行っている状態である。7 + 7 となっている緑色のマス以外のコストとヒューリスティックコストの合計値は 12 となっている。しかし、緑色のマスに関しては、合計値が 14 となる。そのため、このマスよりも合計値が小さくなるマスに対し、引き続き探索を行っていく。

ここで、A*アルゴリズムがダイクストラ法よりも高速に処理ができる理由について述べる。A*アルゴリズムでは、ダイクストラ法と違い、最初に見つけた解が最適解となるため、通常マスすべてのコストを計算する必要はない。A*アルゴリズムは、ゴール地点のコストが計算された時点で探索を終了することができる。そのため、ダイクストラ法よりも探索範囲を縮小することがで

き、処理時間を削減できるため、高速に最短経路を算出することが可能となる。

図 2.7 は、ゴール地点の合計値が算出され、探索が終了した様子を示している。

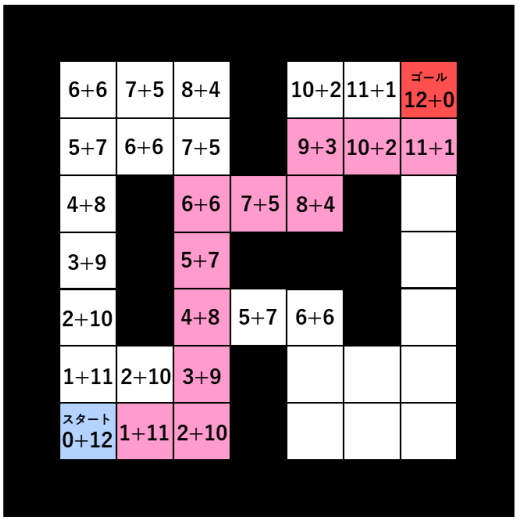

図 2.7　A*アルゴリズムの最短経路算出結果

ゴール地点の合計値が算出された後は、ダイクストラ法のコストが低くなるようにマスを遡った時と同様にマスを遡ることで、最短経路を算出できる。図 2.7 に示されているように、コストとヒューリスティックコストの合計値が計算されていない 9 マス分の処理が行われないため、ダイクストラ法よりも A*アルゴリズムの方が高速に処理を行うことができる。このように、A*アルゴリズムは最短経路を算出する際に、必要にはならないマスの計算を省くことで、処理を簡潔的に行えるようになっている。また、コストマップを構築する上で、合計値を必要以上に記録しておく必要もなくなるため、処理が軽くなる。

2.2.3　Moving Target D* Lite

Moving Target D* Lite[17] は、D* Lite[18] という経路探索手法の改良版である。D* Lite は、最短経路を探索したコストマップで、コストに変化が生じた際に用いられる。D* Lite では、局所的にコストを変更することで、マップ全体に探索をすることなく、高速に最短経路を再探索することができる。

Moving Target D* Lite は、ゴール地点が移動し、同時にコストマップに変化が生じた際に、改良した D* Lite の技術を活用することで高速に最短経路を算出できる。元の D* Lite は、コストマップに変化が生じた際にマップずらすことで処理していた。Moving Target D* Lite では、この処理を行わず、高速に処理できるように D* Lite を改良している。Moving Target D* Lite の説明を、図 2.1 とは違ったコストマップで説明を行う。探索対象マップの構図を図 2.8 に示す。

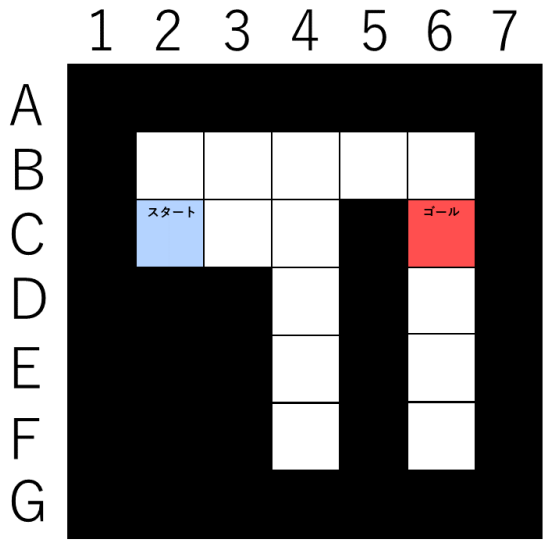

図 2.8　探索対象マップ

次に、図 2.8 のマップで経路探索が行われた様子を図 2.9 に示す。

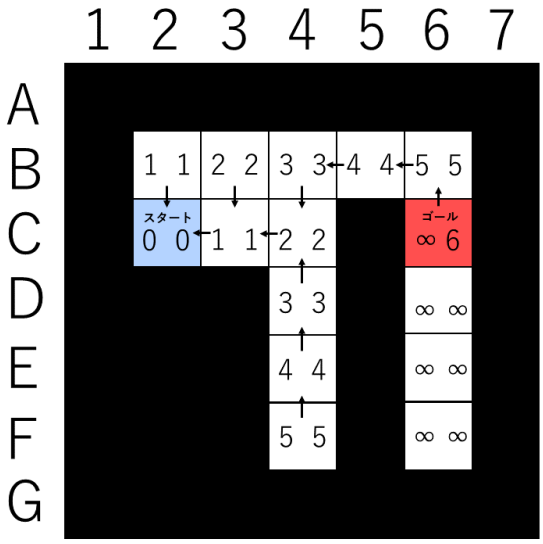

図 2.9　探索結果

図 2.9 に示されている矢印は、経路探索をする際に、自分の親となるマスを記録したものである。親マスとは、対象とするマスのひとつ前のマスのことを指している。図 2.9 の矢印が向いているマスが親のマスとなる。矢印で示されているように、図 2.9 での最短経路は、C2, C3, C4, B4, B5, B6,C6 になる。

図 2.9 に示されている各マスに存在する 2 つの数値は、左側がダイクストラ法によるコスト、右側が right-hand sides[19](RHS) 値と呼ばれるコストを示す。RHS 値は、探索中のマスがスタート地点と同じ場所であれば、そのマスの移動コストをマスに記録する。もし、探索中のマスがスタート地点と違うマスであれば、そのマスの親マスのダイクストラ法のコストと、親マスから現在探索中のマスまでの移動コストの合計値をそのマスの RHS として記録する。また、∞で示されているコストや RHS 値は、それらの計算が行われていない状態を示している。

続いて、コストマップの C5 のマスが通常マスになり、ゴール地点が C6 から D6 へ移動し、ゴール地点が最短経路から外れた状態を図 2.10 に示す。この時、スタート地点は最短経路上を移動し、C2 から C4 へ移動している。

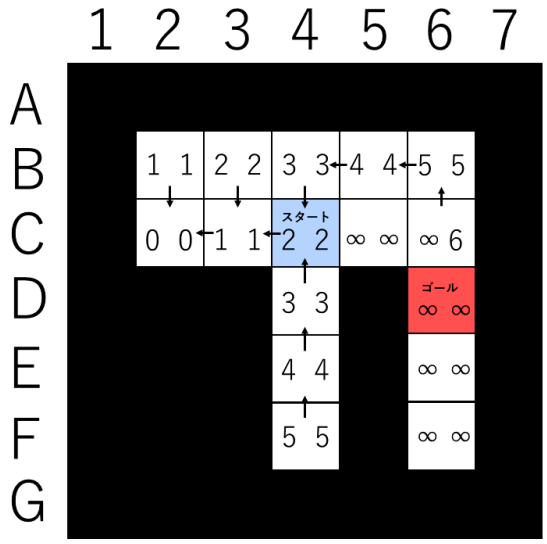

図 2.10　コストマップとゴール地点の変更後

図 2.10 では、スタート地点が C4 に移動したことで、B2, B3, C2, C3 のコストが経路を探索する上で使えなくなってしまう。そのため、一度、それらのマスのダイクストラ法のコストと RHS 値を∞へ更新し、親マスを指す矢印も削除する。また、C5 は C4 を親マスとすることで RHS 値が決まるため、RHS 値を 3 とし、C4 を親マスとして矢印をつける。同様に、B3 と C3 はそれぞれ、B4 と C4 を親マスとすることで RHS 値が決まり、4 と 3 と値を更新できる。その時に、矢印はそれぞれ B4 と C4 を向くようにする。その様子を図 2.11 に示す。

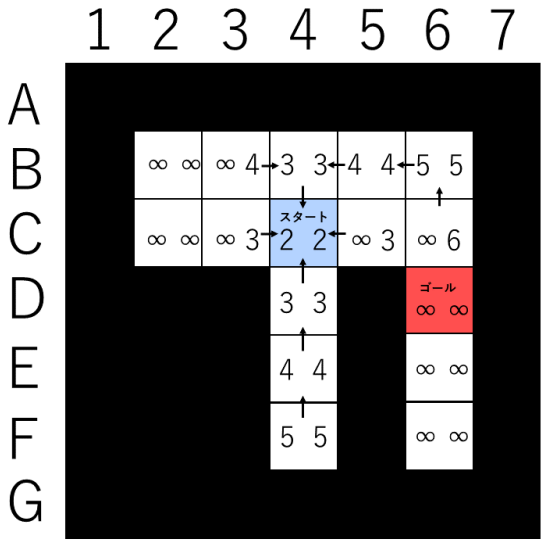

図 2.11　Moving Target D* Lite の初期段階

この後は、ダイクストラ法のコストが∞であり、RHS 値が低いところから順にコストの計算を行っていく。最初に計算されるのが、C3, C5 である。2 つのマスのコストには、3 が記録される。次に、C3, C5 のコストが決まったことで、C2, C6 の RHS 値がどちらも 4 に更新される。このように、コストが確定したマスに隣接していて、RHS 値が小さくなる場合は、更新する。その時に、参照したマスを親マスとして記録する。この作業を繰り返していくと、B3, C2, C6 の RHS 値である 4 が一番小さくなるため、コストが 4 で確定し、B2, D6 の RHS 値に 5 が記録される。その様子を図 2.12 に示す。

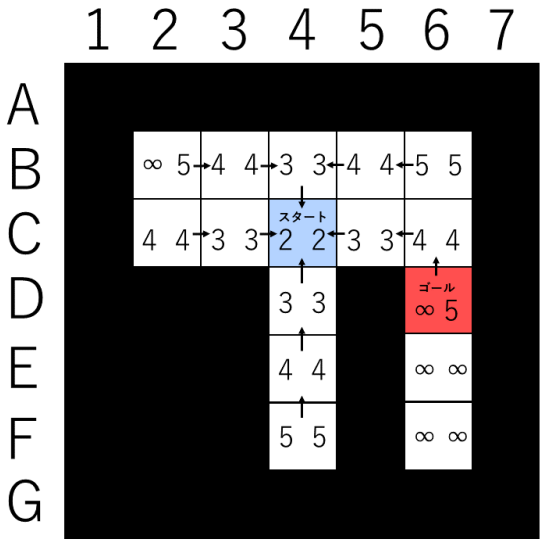

図 2.12　Moving Target D* Lite 処理終了

これでゴール地点のコストが求められたため、探索を終了し、最短経路を算出していき、C4, C5, C6, D6 が最短経路となる。

2.3　現状の問題

ダイクストラ法や A*アルゴリズムでは、一度コストマップを設定すると次回探索する際は、新しく全体のコストを計算することになる。スタート地点がそのままでゴール地点が移動した際に、ゴール地点が既にコストの計算が終わっているマスであれば、新しいゴール地点からコストを遡るだけで経路を探索することができる。しかし、スタート地点が最短経路から外れたマスに移動してしまうと、最短経路を導くことができず、再び全探索をする必要がある。

Moving Target D* Lite は、ゴール地点が最短経路から外れたマスに移動し、コストマップに変化が生じた際にも対応できる手法である。しかし、ゴール地点に向かう対象であるスタート地点が、最短経路として算出された経路から外れてしまうと使えなくなってしまう。また、コストの変更を行う前に前のスタート地点から親子関係が成り立たなくなるマスを削除してからコストの計算をし直すため、削除する量が多くなると計算コストが増えてしまうという問題点がある。

この 3 つの手法は、スタート地点を基準としたコストマップとなるため、スタート地点が移動すると、コストマップの更新が必要となる。そのため、これらの手法では、最短経路から外れてしまった際に、最短経路を導くことができなくなる。つまり、スタート地点とゴール地点が不規則に移動する場合は、最短経路を算出することができない。

第 3 章

提案手法

本章では、提案する経路探索手法について説明をする。3.1 節では、ダイクストラ法の事前データについて、3.2 節では、本手法の流れについて説明する。本研究の手法は、ダイクストラ法による事前データを用いて経路探索を行う手法である。条件として、隣接する通常マスへの移動コストは 1、移動できるマスは上下左右の 4 方向のみとし、ヒューリスティックとしてマンハッタン距離を採用する。また、各通常マスが保有する情報は 5 つある。ダイクストラ法の事前データ、スタート地点が移動した後の追加移動コストを示す Add Move Cost(以下 AMC と呼称する)、ヒューリスティックコスト、前述 3 つの合計コストとマスの親子関係の 5 つである。

3.1 ダイクストラ法の事前データについて

ダイクストラ法の事前データについて説明する。

まず、スタートマスとゴールマス、壁マス、通常マスによって構成されるマップを用意する。この手法では、先述の通り、ダイクストラ法による経路探索のコストマップをもとに行う手法である。今回、手法を説明する上で使用するマップを図 3.1 に示す。

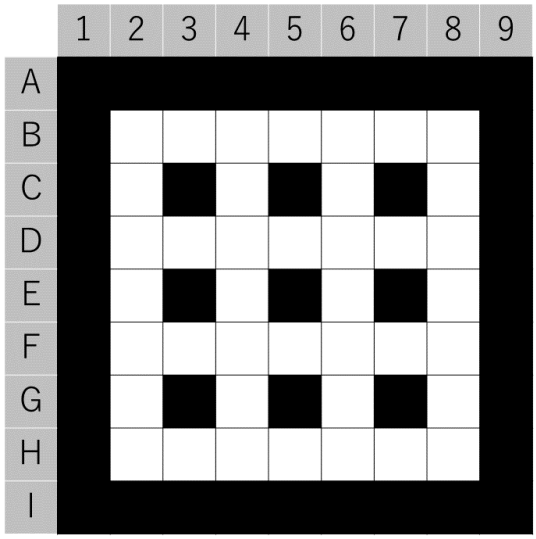

図 3.1　本手法探索対象マップ

図 3.1 に対して、事前にダイクストラ法で経路探索を行う。探索するスタート地点とゴール地点は、それぞれマップの対角に設定する。例えば、左下の H2 がスタート地点の場合、ゴール地点を右上の B8 に設定し、経路探索を行った結果を記録する。この処理を B2,B8,H2,H8 をスタート地点として 4 回行い、それぞれのデータを保有する。図 3.1 の H2 をスタート地点、B8 をゴール地点としてダイクストラ法で経路探索を行ったものを図 3.2 に示す。矢印は、そのマスの親マスを指し示している。第 2 章では、説明を省いたが、A*でも同様に、ゴール地点からスタート地点までマスを遡るために、図 3.2 のように親マスの設定を行っている。

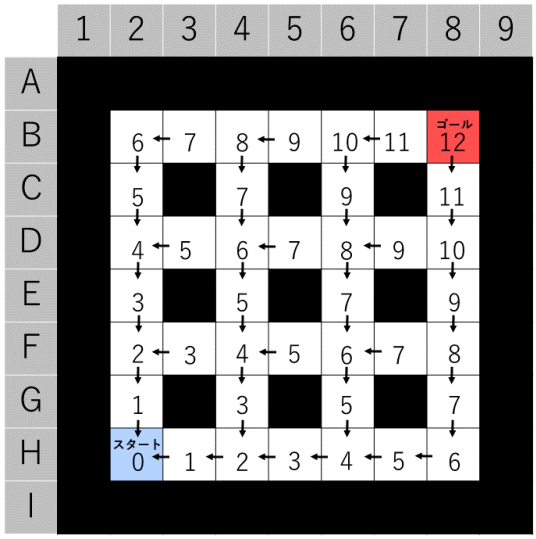

図 3.2　ダイクストラ法の事前データ

続いて、図 3.1 のマップにおいて、スタート地点が H5、ゴール地点が B5 で探索を行ったものを図 3.3 に示す。

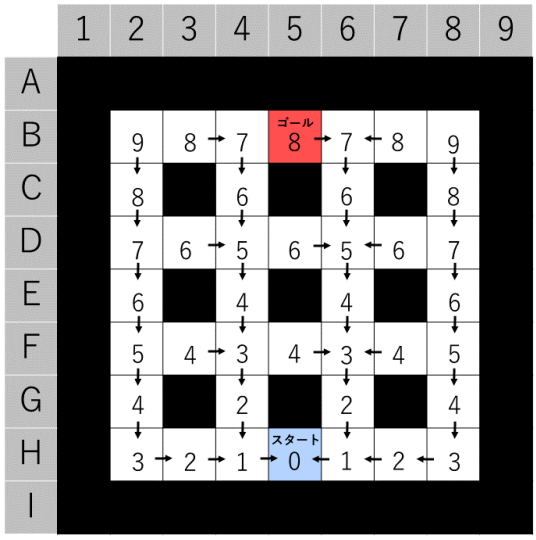

図 3.3　異なるスタート地点とゴール地点の探索コストの様子

図 3.2 のデータをマスの左側に表示し、図 3.3 と同様のスタート地点とゴール地点を設定し、スタート地点の H5 の移動コストを図 3.2 のデータをもとに 3 を基準としてゴール地点の B5 へ探索を行った移動コストを右側に示したものを図 3.4 に示す。

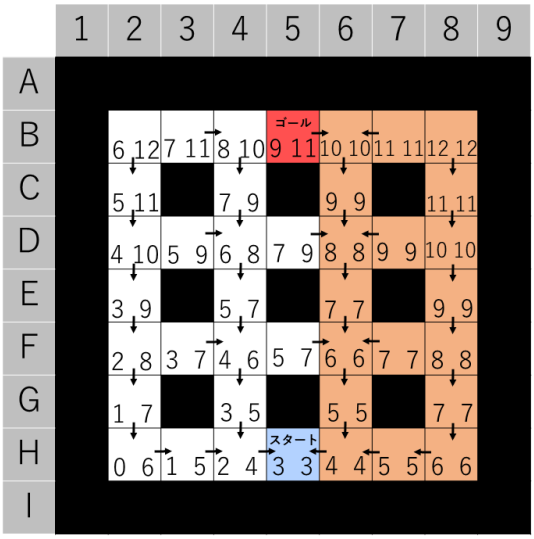

図 3.4　事前データと事前データをもとに移動コストを計算した様子

図 3.2 の一部分は、移動コストを変更することなく、再利用することができる。図 3.4 に示しめしたオレンジ色の B6,B7,B8,C6,C8,D6,D7,D8,E6,E8,F6,F7,F8,G6,G8,H6,H7,H8 のマスと新たなスタートマスは、再利用可能な場所となる。そのようなマスを更新不要マスと呼称する。更新不要マスは、事前データを示す左側のコストと右側に表示した H5 の移動コストを基準とした移動コストが一致しているマスとなっており、このマスはヒューリスティックを更新するのみで探索が終了できる。図 3.3 では、全体的に探索を行うことになるが、図 3.2 の事前データを活用することで、図 3.4 は更新箇所を削減して探索することができる。

3.2　本手法の流れ

本手法の流れについて説明する。

まず最初に、スタート地点とゴール地点が決定したら、スタート地点からゴール地点へのベクトルを算出する。事前データのうち、どれを選択するかは、このベクトルから判断する。ベクトルが指す方向にある角がゴール地点として設定されていた事前データを本手法では、探索の際に採用する。しかし、図 3.4 のような場合は、ベクトルが真上を指しており、H2 から B8 に向かう

データと H8 から B2 に向かうデータの 2 つが候補に挙げられる。その時は、2 つの候補のスタート地点とゴール地点の移動コストの差を出し、より差が大きい方をデータとして採用し、同じ場合はどちらを選択しても問題はない。差が小さい方のデータは、差が大きい方のデータに比べ、更新すべきマスが多くなる可能性が非常に高いからだ。

次に、探索を行っていく。探索を行う際、更新不要マスでは、ヒューリスティックのコストを計算するだけで処理を終了する。それ以外のマスに関しては、A*アルゴリズム同様に移動コストとヒューリスティックのコストを算出する。更新不要マスとそれ以外のマスの選別方法は、探索をする際に参照中のマスとその周囲のまだ探索を行っていないマスに対して親子関係があるかを判断し探索を行う。参照中のマスの移動コストを S, 探索対象のマスの移動コストを T, 参照中のマスの AMC を P, 探索対象のマスの AMC を Q, 参照中のマスから探索対象のマスへの移動コストを C とする。もしも、探索対象のマスの親が参照中のマスであり、P が 0 であれば、ヒューリスティックコストを計算し、処理を終了する。それ以外の場合は、移動コストが間違っている可能性があるため、式 (3.1) の計算を行う。

$$Q = S + P + C - T. \tag{3.1}$$

式 (3.1) の値と T を移動コストとし、ヒューリスティックコストを求め、親のマスを参照中のマスに設定しなおし、探索対象のマスにおける処理は終了する。

スタート地点を H5, ゴール地点を B5 として本手法で探索を行っていく様子を図をもとに説明する。事前データはスタート地点が H2 でゴール地点が B8 の探索データを利用する。スタート地点である H5 は、更新不要マスであるため、ヒューリスティックを計算し、合計コストを算出する。次にその周りのマスに対して、上記の処理を行う。H4,H5,H6 の計算が終了した状態を図 3.5 に示す。

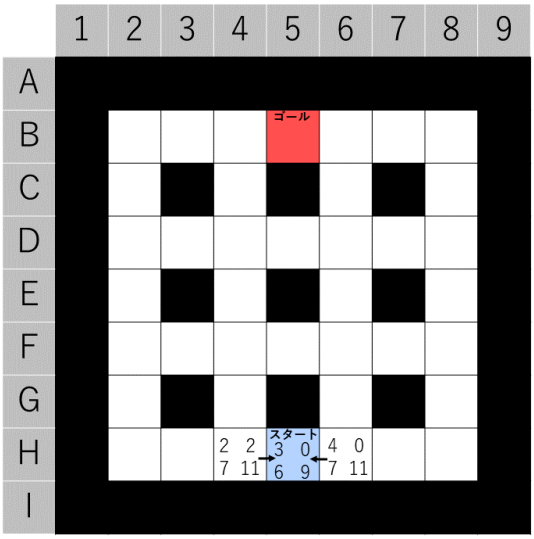

図 3.5　本手法の初期段階

各マスの左上が事前データの移動コスト、右上が追加移動コストの AMC、左下がヒューリスティックコスト、右下がこれらの合計コストを示している。H4 は、親子関係が成り立たないため、式 (3.1) による計算を行い、$Q = 3 + 0 + 1 - 2$ より、$Q = 2$ となる。H4 のヒューリスティックは 7 となり、合計コストは $2 + 2 + 7 = 11$ となる。また、親子関係が成り立たないため、H4 の親マスを H5 とする。H6 は、親子関係が成り立つため、ヒューリスティックの 7 と移動コストを合計し、11 となる。同様の手順でゴール地点まで処理を行い、最短経路を算出し、ピンク色のマスで示したものが図 3.6 である。ゴール地点まで探索が行われたら、ゴール地点から親マスをスタート地点まで遡ることで、最短経路を導き出すことができる。

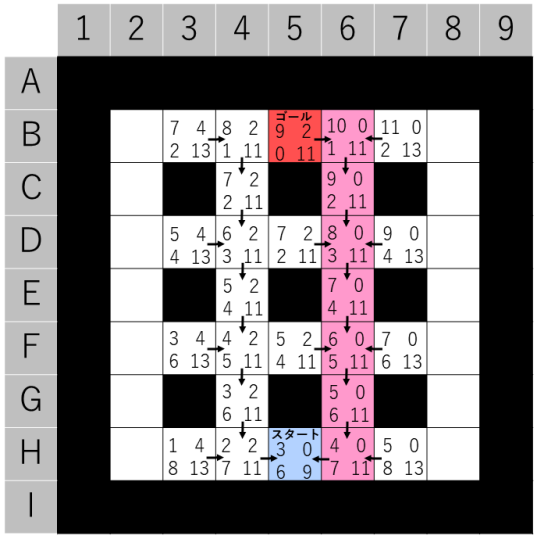

図 3.6　本手法の探索終了状態

図 3.4 に示したオレンジ色の更新不要マスの AMC は 0 となっており、図 3.6 の約半分ほど移動コストの計算を省くことができる。また、A*と本手法の違いは、A*では再探索の際に前回の探索の移動コスト、ヒューリスヒックコスト、合計コスト、親子関係のすべてを初期化しなおす必要がある。しかし、本手法では、AMC、ヒューリスティックコスト、合計コストを初期化するだけで成り立つため、データの初期化する量を減らすことができる。

第 4 章

提案手法の検証

本章では、本手法の実験・分析結果、考察について述べる。4.1 節では、実験方法について説明し、4.2 節では、実験の結果について説明する。最後に 4.3 節では、考察を述べる。

4.1 本手法と既存手法の比較実験

本研究は、スタート地点とゴール地点がそれぞれ移動し、再探索を行う際の処理速度の向上を目的としている。そこで、探索する際にかかる処理速度を計測し、既存手法と比較実験を行う。

この実験では、Fine Kernel ToolKit[20] を用いて、プログラムを作成した。作成したプログラムは、3 つある。A*アルゴリズム、Moving Target D* Lite、本研究が提案する手法の 3 つでの経路探索を実装している。実行時のスタート地点とゴール地点をそれぞれ、旧スタート地点、旧ゴール地点とし、移動後の各地点を新スタート地点、新ゴール地点と呼ぶ。また、それぞれの最短経路を旧最短経路と新最短経路とする。このプログラムを実行時の旧スタート地点と旧ゴール地点の旧最短経路を図 4.1 に、各手法の実行後の新スタート地点と新ゴール地点の新最短経路を図 4.2 に示す。

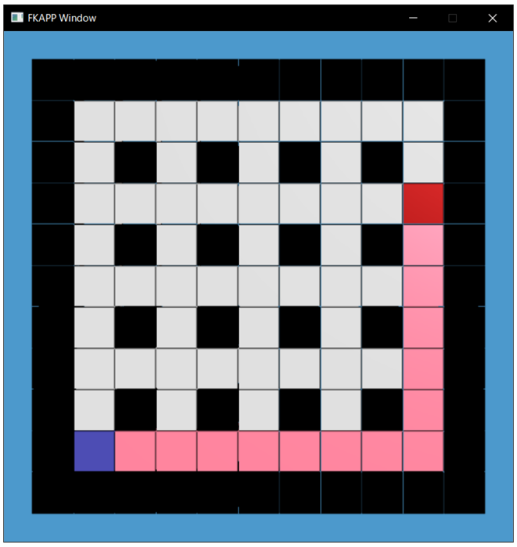

図 4.1　プログラム実行時の旧スタート地点と旧ゴール地点の旧最短経路の探索結果

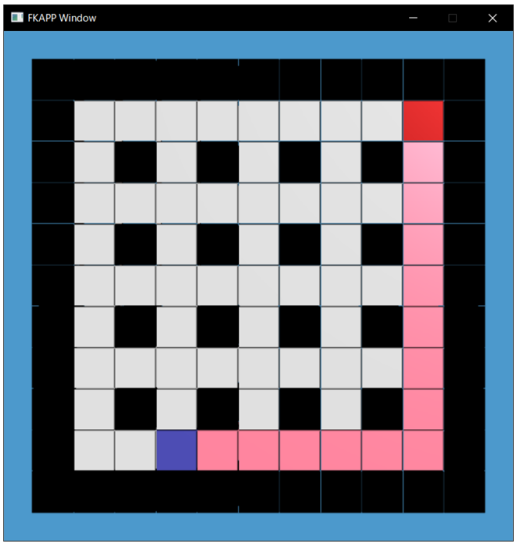

図 4.2　各手法の実行後の新スタート地点と新ゴール地点の新最短経路の探索結果

マップには様々な特徴があると判断し、マップの特徴と新スタート地点と新ゴール地点の位置に着目して実験をする。マップの特徴として、迷路のように入り組んだ構造のものと、山道のように大きな一本道にいくつかの脇道がある構造、建物が規則的に等間隔で並ぶ構造の 3 種類を想定する。迷路のマップを図 4.3 に、ほぼ一本道のマップを図 4.4 に、規則的なマップを図 4.5 に示す。これは、実際のオープンワールドゲームにも含まれている構造である。

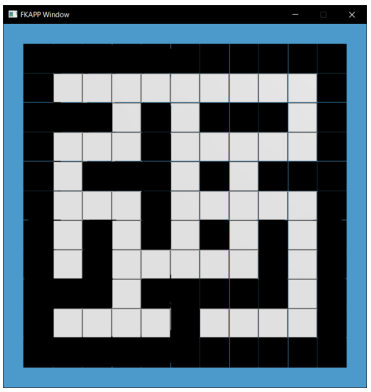

図 4.3　迷路マップ

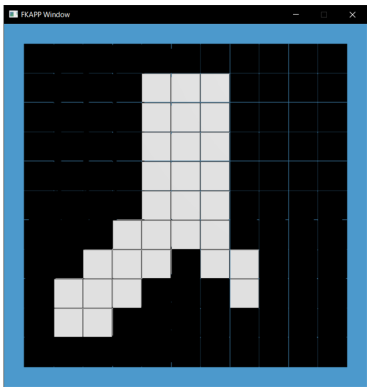

図 4.4　ほぼ一本道のマップ

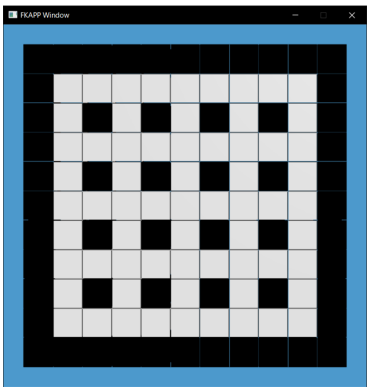

図 4.5　規則的なマップ

この実験における実際のマップのサイズは、マスを縦横に 5001 × 5001 で配置したもので行う。

次にスタート地点とゴール地点について考える。1 つ目が、新スタート地点がプログラム実行時の旧スタート地点と旧ゴール地点の最短経路上を移動し、新ゴール地点が最短経路外に移動した場合である。2 つ目が、新スタート地点も新ゴール地点も最短経路外に移動した場合である。

4.2　比較実験の結果

新スタート地点が旧最短経路上を移動し、新ゴール地点が旧最短経路外に移動したときの結果を表 4.1 に、新スタート地点と新ゴール地点が旧最短経路外に移動した時の結果を表 4.2 に示す。

表 4.1　新スタート地点が旧最短経路上を移動した場合 (ms)

	A*	MTD* Lite	本手法
迷路マップ	4.886	2.10000000002708E-06	5.301
ほぼ一本道のマップ	1.670	0.002	1.445
規則的なマップ	4.469	0.299	2.849

表 4.2　新スタート地点が旧最短経路外に移動した場合 (ms)

	A*	MTD* Lite	本手法
迷路マップ	4.738	—	6.643
ほぼ一本道のマップ	1.674	—	1.481
規則的なマップ	4.437	—	2.924

新スタート地点が旧最短経路上を移動する場合は、本手法よりも、Moving Target D* Lite の方が処理速度が高速であることがわかる。しかし、新スタート地点が旧最短経路外に移動する場合は、Moving Target D* Lite は処理ができず、本手法が A*よりも速く処理することができる。

4.3　考察

まず、表 4.1 を見ると、新スタート地点が旧最短経路上を移動する場合は、いづれのマップにおいても Moving Target D* Lite が高速に処理できることがわかる。本手法は、迷路マップでは、A*よりも処理が遅くなっているが、他のマップでは、わずかに速く処理することができる。よって、新スタート地点が旧最短経路上を移動した場合は、Moving Target D* Lite を用いることで高速に処理することができる。次に、表 4.2 を見ると、新スタート地点が旧最短経路外に移動する場合は、Moving Target D* Lite は、処理ができないため、迷路マップを除いて、本手法がわずかに速く処理できることがわかる。2 つの結果から、迷路マップにおいては、新スタート地点の位置に関係なく、本手法よりも A*の方が速く処理できるため、本手法は迷路マップには不向きであると判断できる。

以上の結果から、3 つの探索手法には特徴があることがわかる。Moving Target D* Lite は、

A*や本手法よりも高速に処理できるが、新スタート地点が旧最短経路上を移動する場合のみ有効である。また、A*は汎用的で、迷路マップにおいては本手法よりも速く処理することができるが、他のマップでは処理が遅くなる。最後に本手法は、新スタート地点が旧最短経路外に移動する場合の迷路マップ以外で、有用であると言える。A*と比べ若干速くなるが、A*と同じ量のマスを探索する必要があるため、Moving Target D* Lite ほどの処理削減とはならなかった。

本手法は、オープンワールドゲームにおける広大なマップでの探索手法を想定していたが、A*との差もわずかであり、手法の準備にも手間がかかるため、まだまだ実践で使用するには困難であると言える。

第 5 章

まとめ

本研究では、従来の経路探索アルゴリズムである、A*アルゴリズム、Moving Target D* Lite と本手法の 3 つを比較した。比較実験から、マップの形状と新スタート地点、新ゴール地点の 3 つの要素によって、得意不得意があることがわかった。本研究で提案する手法は、マップが迷路のように複雑ではなく、一本道や障害物が点々と存在する単純な構造であり、かつ、新スタート地点と新ゴール地点の両方が旧最短経路外に移動した際に、有用であることが判明した。新スタート地点が旧最短経路上に移動し、新ゴール地点が旧最短経路外に移動した際には、Moving Target D* Lite の方が、本手法よりも高速に処理することができる。しかし、Moving Target D* Lite が処理できない新スタート地点と新ゴール地点が旧最短経路外に移動した際に、汎用的な A*アルゴリズムよりわずかだが速く処理できることがわかった。

このことから、迷路のような複雑ではないマップにおいて、アルゴリズムを使い分け、新スタート地点が旧最短経路上を移動する場合は、Moving Target D* Lite による経路探索を行い、新スタート地点と新ゴール地点が旧最短経路外に移動する場合は、本手法による経路探索を行うことで、より速く処理することができると考えた。

本手法では、探索の事前データによる処理速度の高速化を目指した。しかし、上述のように、A*との差がわずかであり、迷路や行き止まりの多い複雑なマップでは、最適な事前データを用意することが困難であること、マップ全体のデータをいくつか保有する必要があるため、メモリの問題も考えられる。そこで、より汎用的な事前データや処理方法などを考える必要がある。

また、新スタート地点と新ゴール地点の位置に応じて処理の変更を効率的に行えるようにするアルゴリズムの切り替えの処理や隣接マスへの移動コストが 1 以外の場合での処理も重要となる。

謝辞

本研究、本論文のご指導いただいた、渡辺先生、阿部先生に心より感謝いたします。また、相談やサポートをしてくださった先輩方や友人たちにも深く感謝をしております。誠にありがとうございました。

参考文献

[1] 三宅陽一郎. 人工知能の作り方 - 「おもしろい」ゲーム AI はいかにして動くのか. 技術評論社, 2016.

[2] David M. Bourg and Glenn Seemann. ゲーム開発者のための AI 入門. O'Reilly Japan, 2005.

[3] 神崎洋治. ゼロからわかる AI 時代のロボットの仕組みと活用. SB クリエイティブ, 2017.

[4] New Energy and Industrial Technology Development Organization. ロボット・AI | NEDO. https://www.nedo.go.jp/activities/introduction_100017.html. 参照: 2021.7.12.

[5] 角本繁, Kisi-W コンソーシアム. カーナビゲーションシステム-公開型データ構造 KIWI とその利用方法-. 共立出版, 2003.

[6] 2021 Yahoo Japan Corporation. Yahoo! 路線情報のスマートフォンアプリ「Yahoo!乗換案内」. https://transit.yahoo.co.jp/smartphone/app/. 参照: 2021.4.7.

[7] A&A CO. Sim Tread 2021. https://www.aanda.co.jp/products/simtread/index.html. 参照: 2021.7.11.

[8] 池田裕太郎, 池田心. アクションゲームにおける特定のプレイヤの特徴を摸倣する AI プレイヤの作成. 情報処理学会 研究報告ゲーム情報学 (GI), Vol. 2020-GI-43, No. 9, pp. 1–8, 2020.

[9] 佐藤直之, 池田心. Influence Map を用いた経路探索による人間らしい弾避けのシューティングゲーム. ゲームプログラミングワークショップ 2016 論文集, Vol. 2016, pp. 57–64, 2016.

[10] 中宮正樹, 岸野泰恵, 寺田努, 西尾章治郎. コストマップを用いた移動型センサノードの経路探索手法. 情報処理学会論文誌, Vol. 49, No. 3, pp. 1374–1386, 2008.

[11] 杉山大貴, 小野貴彦. データベースを利用した救急車の経路探索の高速化. 中国四国支部総

会・講演会　講演論文集, Vol. 57, No. 1015, 2019.

[12] E.W.Dijkstra. A note on two problems in connexion with graphics. *Numerische Mathematik*, Vol. 1, pp. 269–271, 1959.

[13] B. コルテ, J. フィーゲン. 組み合わせ最適化：理論とアルゴリズム. シュプリンガー・ジャパン株式会社, 2009.

[14] 2021 Yahoo Japan Corporation. Yahoo! 路線情報 : 乗換案内、時刻表. `https://transit.yahoo.co.jp/`. 参照: 2021.4.7.

[15] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimal Cost Paths. *IEEE transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107, 1968.

[16] 2016 Naoto Mukai. 探索アルゴリズム・ヒューリスティック探索. `https://mukai-lab.info/pages/classes/artificial_intelligence/chapter7/`. 参照: 2021.4.7.

[17] Xiaoxun Sun, William Yeoh, and Sven Koenig. Moving Target D* Lite. *9th International Conference on Autonomous Agents and Multiagent Systems(AAMAS)*, Vol. 1, pp. 67–74, 2010.

[18] Sven Koenig and Maxim Likhachev. D* Lite. *AAAI Conference of Artificial Intelligence (AAAI)*, pp. 476–483, 2002.

[19] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong Planning A*. *Artificial Intelligence*, Vol. 155, pp. 93–146, 2004.

[20] 1997-2020 Fine Kernel Project (fk@gamescience). Fine Kernel ToolKit Top Page. `https://gamescience.jp/FK/`. 参照: 2021.6.23.