芸術科学会論文誌 Vol. 21, No. 2, pp. 111 – 122

事前探索データを用いた経路探索手法

吉田涼真 1)(学生会員)　阿部雅樹 2)(正会員)　渡辺大地 2)(正会員)

1) 東京工科大学大学院バイオ・情報メディア研究科 2) 東京工科大学メディア学部

Path Finding Method Using Pre-search Data

Ryoma Yoshida[1)]　**Masaki Abe**[2)]　**Taichi Watanabe**[2)]

1) Graduate School of Bionics, Computer and Media Science, Tokyo University of Technology
2) School of Media Science, Tokyo University of Technology

g312102830@edu.teu.ac.jp, abemsk@edu.teu.ac.jp, earth@gamescience.jp

概要

経路探索は，ある地点から目的地点への経路を算出するものである．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，現在の最短経路上から外れた位置に新たな目標地点を設定した際，経路探索を再び行う必要がある．経路探索手法 Moving Target D* Lite 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．しかし，この手法は，現在の最短経路上から外れた位置に新たなスタート地点を設定すると成り立たなくなってしまう．本研究は，新規スタート地点や新規ゴール地点を現在の最短経路外に設定した際に先行手法よりも速く新規最短経路を算出可能な探索手法を提案する．提案手法では，ダイクストラ法による探索情報を事前探索データとして複数記録しておき，再探索時にその中から最適なデータを採用し，計算量を削減する．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現在の最短経路上から外れた位置に新規スタート地点を設定する場合，有用であることが判明した．

Abstract

Pathfinding is the calculation of a path from a point to a destination point. The main methods are Dijkstra method and A* algorithm. In these two methods, when a new target point is set outside the current shortest path, the path search needs to be performed again. The path finding method Moving Target D* Lite can find the shortest path fast under such conditions and when changes occur in the map. However, this method is no longer viable when a new starting point is set at a location that is off the current shortest path. In this study, we propose a search method that can calculate a new shortest path faster than the previous method when a new start point or a new goal point is set outside the current shortest path. In the proposed method, the search information by Dijkstra method is recorded as multiple pre-search data, and the optimal pre-search data is adopted from multiple pre-search data . For squares whose cost needs to be updated based on the pre-search data, the cost is saved as an additional movement cost and the search is performed. As a result of verifying this method, we found that it is useful when setting a new starting point at a location that is off the current shortest path for a map that has almost a single path like a mountain road or regular obstacles like a building district.

1　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] などに用いられる．経路探索を実現する主な手法として，ダイクストラ法 [6] や A*アルゴリズム [7] が挙げられる．

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，探索の出発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行われることはない．しかし，ゲーム 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．一例として，アクションゲームにおける NPC(ノンプレイヤーキャラクター) がプレイヤーの操作するキャラクターに向かって移動する状況下がある．NPC は経路探索の結果をもとに目的地点であるプレイヤー操作キャラクターへ移動をするが，プレイヤー操作キャラクターは頻繁に位置が変わるため，NPC にとってゴール地点は頻繁に変更が行われる．また，NPC が目標地点へ移動している最中に，経路上に木などが倒れて障害物となる，地面が崩れて道として機能しなくなる等といったゲーム特有のギミックが発生した場合，NPC は初めに算出した目標地点への経路外に移動する必要が生じる．経路外に移動した後に，再度その地点からの経路探索が必要となる．

その際に，更新したスタート地点やゴール地点が一度計算した最短経路上にそれぞれの新規地点が設定されている場合は，経路情報をそのまま利用できる．しかし，それぞれの新規地点が現在の最短経路上から外れた位置にある場合，一から探索を行う必要がある．

本研究の目的は，新規スタート地点や新規ゴール地点を現在の最短経路外に設定した際に，事前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提案を行うことである．

我々は，本研究の初期成果を NICOGRAPH2021 にて発表し [8]，事前データ群の差違による影響についてが不明である旨の指摘を受けた．本論文では，事前データ群が異なる場合として，事前探索データ群を対角線上に探索を行ったデータとは別に，対辺上に探索を行ったデータを用いて検証を実施した．これら 2 つの事前探索データ群による処理時間の違いを示すために，それぞれ 3 つのマップに対しての比較実験を行った．

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現在の最短経路上から外れた位置に新規スタート地点を設定する場合，有用であることが判明した．

2　本論文で取り扱うマップについて

本論文では，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 1 は，格子状に区切ったマップを示している．

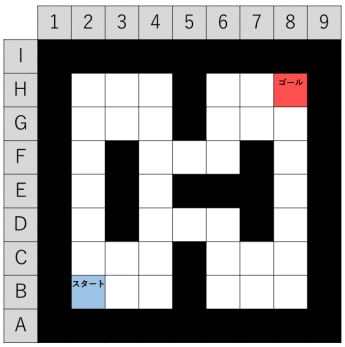

図 1　格子状のマップ

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．また，本研究では一方通行の経路は考慮せず，全ての経路が双方向に移動可能なものとする．

3　経路探索の先行研究

3.1　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から順にスタート地点からの到達コストを計算していく．到達コストとは，探索の開始地点となるマスから対象となるマスへ移動する際に必要となるコストのことを指す．まず，スタート地点の到達コストを 0 とし，

その後その周りにある通常マスから到達コストを計算していく．その様子を図 2 に示す．

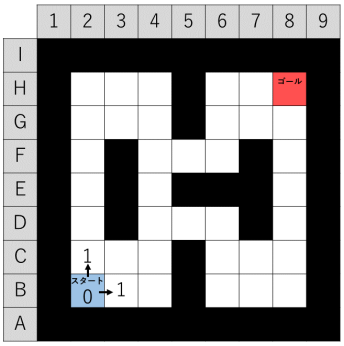

図 2　探索の初期段階

図 2 で示している矢印は，対象マスと隣接するマスの経路を示している．矢印の向きは経路の向きを示すものであり，終点側の到達コストは始点側の到達コストよりも高いものとなる．

次に，コスト 1 を割り振っ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．この作業は，ゴール地点の計算が終了した時点で処理を終了する．この計算結果から最短経路を算出する．最短経路が算出された様子を図 3 に示す．最短経路となるマスはピンク色で示す．

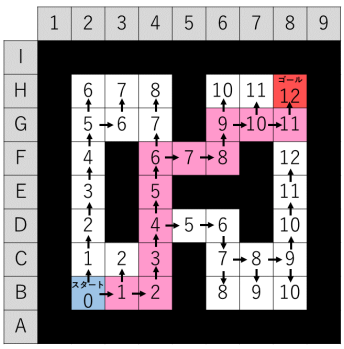

図 3　最短経路の算出結果

ダイクストラ法では，ゴールよりコスト値が常に減少するように辿ることで最短経路の算出を行う．カーナビなどでは，料金・時間などの目的に合わせた到達コストを使用することで最適経路の算出を行う．ダイクストラ法によって算出される経路は正確だが，後発の手法に比べマップの広範囲に対して探索を行うため，処理時間が大幅にかかってしまう．

3.2　A*アルゴリズム

A*アルゴリズムは，スタート地点からゴール地点まで再帰的に到達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と異なる，ヒューリスティックコスト [9] と呼ばれるコストを用いる．ヒューリスティックコストは，ある地点からゴール地点までの最小コストの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速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多い．本研究では，マンハッタン距離を使用する．そのため，ヒューリスティックの値を h としたとき，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 (1) となる．今回は，スタート地点の座標を $(0, 0)$ としたとき，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．

$$h = |X - x| + |Y - y|. \tag{1}$$

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到達コストを求め，各マスの合計コストが低くなるマスから計算をする．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という特徴があるため，到達コストを算出するマスの個数が減少し，結果的に算出時間の短縮が期待できる．

図 4 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．左の数値がダイクストラ法による到達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．また，ピンク色で示したマスが最短経路を示している．

図 4 では，空白のマスがある．空白のマスは到達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マスを示している．この 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，不要なマスの計算を省くことで，処理を簡潔に行えるようになっている．

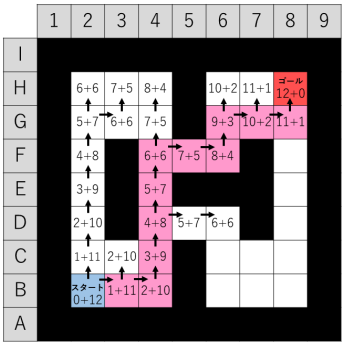

図 4　A*アルゴリズムの最短経路算出結果

3.3　Moving Target D* Lite

Moving Target D* Lite[10] は，経路探索手法 D* Lite[11] 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る．コストマップとは，経路を算出するために必要となるコストの記録のことである．

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，高速な最短経路算出を実現できるように D* Lite の技術を改良したものである．図 5 に示しているのは，Moving Target D* Lite で経路探索をした様子である．

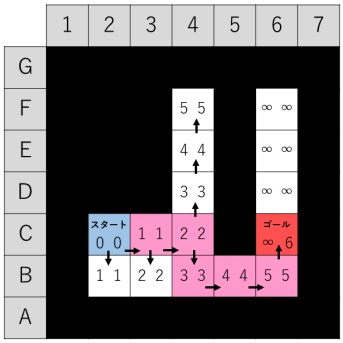

図 5　探索結果

図 5 に示し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到達コスト，右側が right-hand sides[12](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，そのマスの到達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到達コストと，親マスから現在探索中のマスまでの到達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場合での初期値として設定する．

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，最短経路上から外れた位置にゴール地点を設定した時の経路探索を考える．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．スタート地点が C4 に移動したため，前のスタート地点である C2 のダイクストラ法の到達コストを ∞ に更新し，親マスを指す矢印も削除する．C2 を親マスとしていたマスのダイクストラ法のコストと RHS 値を順に更新した様子を図 6 に示す．

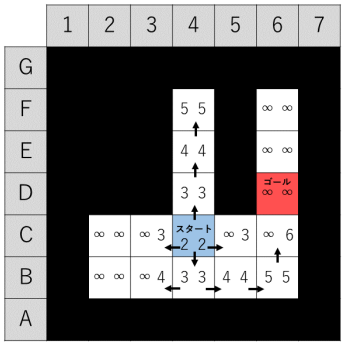

図 6　Moving Target D* Lite の初期段階

この後は，ダイクストラ法の到達コストが ∞ 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．探索が終了した様子を図 7 に示す．

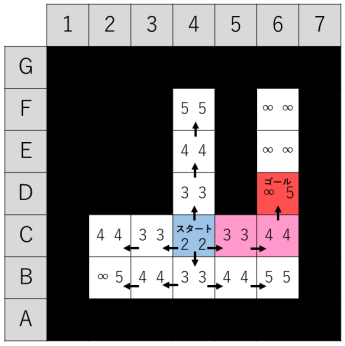

図 7　Moving Target D* Lite 処理終了

3.4　先行研究の問題

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，新しく一からコストを計算することになる．

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，最短経路外にスタート地点を設定すると使えなくなってしまう．ゲームAIのような分野では，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なるため，先行研究は不向きである．

4　提案手法

本手法では，ダイクストラ法やA*アルゴリズムとは違い，事前に探索を行ったデータを複数用意しておく．そのデータのことを事前探索データと呼び，そのまとまりを事前探索データ群と呼称する．現在探索しようとしている状況に一番近い事前探索データを選択し，そのデータをもとに探索を行う．これにより，探索処理を減らして経路を算出することができる．

本手法の処理を大きく分けると，以下のようになる．

1. マップ情報より事前探索データを複数用意する．これはマップ構成に大きな変化がない限り，最初に実行するのみでよい．
2. スタート地点とゴール地点が設定された際に，使用する事前探索データを選択する．
3. マップ内のマスを更新必要マスと更新不要マスに分ける．
4. 各マスのコスト情報を更新する．
5. 最短経路を抽出する．

本章では，各項目の処理の詳細を述べる．

4.1　事前探索データ群の作成

本節では，本手法の最初の手順である事前探索データ群の作成について述べる．本手法は，ダイクストラ法による経路探索のコストマップを事前探索データとして用意する．

今回，手法を説明する上で使用するマップを図8に示す．

図8に対して，事前にダイクストラ法で経路探索を行う．本研究の事前探索データは，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し，対角線上に探索を行ったものとした．例として，左下のB2がスタート地点の場合，ゴール地点を右上のH8に設定し，経路探索を行った結果を図9に示す．

本研究では，事前探索データ群として，B2,B8,H2,H8をそれぞれスタート地点として対角線上に探索した事前探索データを採用した．対角線上に探索を行った事前探索データ群を図10に示す．

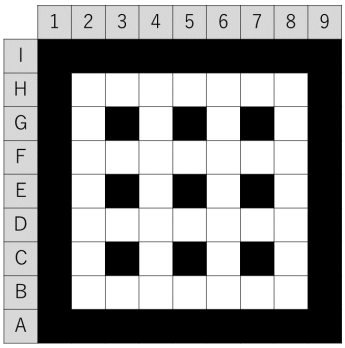

図8　本手法探索対象マップ

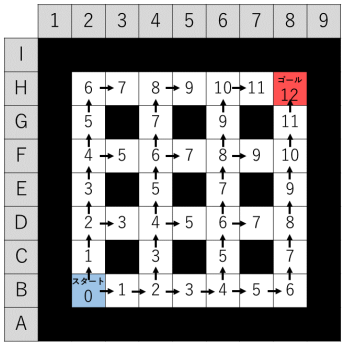

図9　ダイクストラ法の事前探索データ

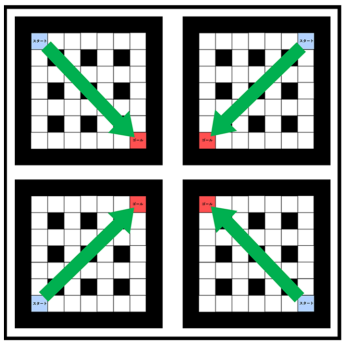
図10　対角線上に探索を行った事前探索データ群

4.2　事前探索データの選択

本節では，事前探索データ群の中から1つの事前探索データを選択する方法について説明する．

まず，n 個の探索データによる事前探索データ群 $\{M_1, M_2, \ldots, M_n\}$ において，マップ M_i のスタート地点位置とゴール地点位置を $\mathbf{S}_i, \mathbf{G}_i$ とし，$\mathbf{V}_i = \frac{\mathbf{G}_i - \mathbf{S}_i}{|\mathbf{G}_i - \mathbf{S}_i|}$ とする．

次に，探索対象となるマップのスタート地点とゴール

地点が決定したら，スタート地点位置 $\mathbf{S}$ からゴール地点位置 $\mathbf{G}$ への方向ベクトル $\mathbf{V} = \frac{\mathbf{G}-\mathbf{S}}{|\mathbf{G}-\mathbf{S}|}$ を算出する．スタート地点が D3，ゴール地点が H6 の時の $\mathbf{V}$ の様子を図 11 に示す．

どの事前探索データを選択するかは，以下の式 (2) により決定する．

$$\arg\max_{i\in\{1\ldots n\}} \mathbf{V}\cdot\mathbf{V}_i \tag{2}$$

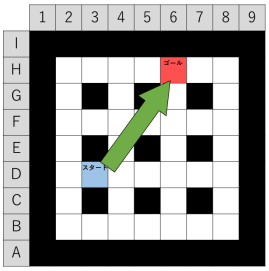

図 11　スタート地点を D3，ゴール地点を H6 としたときの $\mathbf{V}$ の様子

図 11 の場合，図 10 に示した事前探索データ群の中で，左下にある B2 をスタート地点としたときの事前探索データでの式 (2) の値がもっとも大きくなるため，図 10 の左下の事前探索データを採用することとなる．

しかし，図 12 に示すように，スタート地点が B5，ゴール地点が H5 のような場合，$\mathbf{V}$ が真上を指している．そのため，図 13 に示す，B2 から H8 に向かうデータと B8 から H2 に向かうデータの 2 つが事前探索データの候補に挙がる．

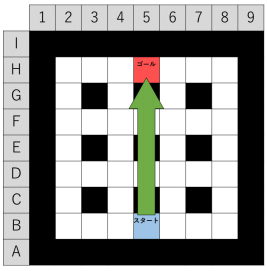

図 12　スタート地点を B5，ゴール地点を H5 としたときの $\mathbf{V}$ の様子

このような場合，2 つの候補のスタート地点とゴール地点の到達コストの差を出し，より差が大きい方を事前探索データとして採用する．これは，差が大きい方のデータの方が，差が小さい方のデータに比べ，更新すべきマスが少なくなる可能性が高いからである．もし，同じ場合はどちらを選択しても問題はない．

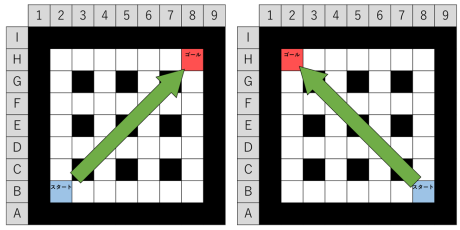

図 13　候補として挙がる 2 つの事前探索データ

図 12 のケースでは，B2 をスタート地点とする事前探索データの H8 と B2 の到達コストの差と B8 をスタート地点とする事前探索データの H2 と B8 の到達コストの差を比較するが，差の値は同じになり，どちらの事前探索データでも問題はないため，B2 をスタート地点とする事前探索データを採用する．

4.3　更新必要マスと更新不要マスの判定

本手法の高速化の主要因は，コストの再計算が必要となるマスと不要なマスに分類し，不要なマスによる再計算を削減できることによるものである．そのため，更新が必要か不要かを分類することは重要な処理となる．本節では，まずその概要を例示によって解説し，具体的な分類方法について述べる．

まず，事前探索データとして，事前探索データ群の図 10 の左下と同じ図 9 のデータを用いることとする．図 14 の各マスの左側は到達コストを記したものである．次に探索条件として，スタート地点を B5，ゴール地点を H5 に設定する．先行手法ではスタート地点の到達コストを 0 とするのが通例であるが，本手法では図 9 における B5 のコストを事前探索データのコスト値である 3 に置き換えて探索を行う．スタート地点のコストを 3 として探索した各マスの数値をマスの右側に記し，各マスに 2 つの値を示したものを図 14 に示す．また，図 14 に示した矢印の向きは，スタート地点のコストを 3 として探索した時のものである．

事前探索データの数値である左側と，スタート地点のコストを事前探索データの値に置き換えて探索した右側の値が同値となる箇所は，事前探索データとして再利用可能な箇所となる．図 14 に示したオレンジ色の B6～8, C6, C8, D6～8, E6, E8, F6～8, G6, G8, H6～8 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」とし，それ以外を「更新必要マス」と定義する．図 14 では，全てのマスにお

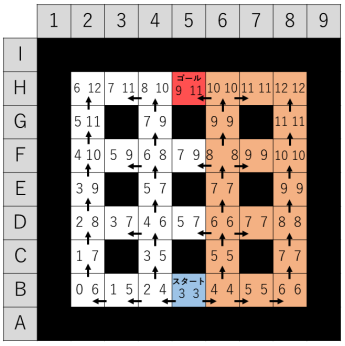

図 14　事前探索データと事前探索データをもとに到達コストを計算した様子

ける結果を示しているが，実際は更新不要マスを探索時に適宜判別し，探索を行う．先行手法による探索では全体的に探索を行うことになるが，図 9 の事前探索データを活用することで，図 14 は更新箇所を削減して探索することができる．

ここで，事前探索による到達コスト値から新たに算出した到達コスト値を引いた値を「追加コスト」と定義する．例として，図 14 の B4 をもとに説明する．B4 は本来，到達コストとして値 4 が適切であるが，値 2 が記録してある．そのため，事前探索データの B4 に不足している値 2 を追加コストとして記録することで，不足している値を補完することができる．追加コストは，次節で述べるように事前に算出して各マスに設定する値であるため，「更新不要マス」と「更新必要マス」の判定は，追加コストの値が 0 かどうかで判断できる．

4.4　各マスの情報設定

前節で述べた事前探索データを元に，各マスに対して以下の 4 つの情報を設定する．

- スタート地点が移動した後の追加コスト
- ヒューリスティックコスト
- 合計コスト
- 前マスにあたるマス

ここで,「合計コスト」とは追加コスト，ヒューリスティックコスト，事前探索データの同一位置マスでの到達コストの 3 つのコストの合計値のことである．また，「前マス」とは，処理対象となるマスに隣接するマスのうち，注目マスより到達コストが低いマスのことと定義する．条件に合うマスが複数ある場合は，そのうちの任意の 1 つとする．

図 12 の条件で説明を行う．処理の順序として，スタート地点およびスタート地点に隣接するマスについて順に注目していく．

まず初めに，スタート地点について述べる．スタート地点は更新不要マスに属しているため，追加コストは 0 とする．次に，ヒューリスティックコストは事前探索データには存在しないため，式 (1) を用いて算出する．到達コストは事前探索データの値をそのまま利用し，到達コストとヒューリスティックコストの合計値を合計コストに設定する．前マスについては，スタート地点は大元のマスなので「無し」とする．

続いて，スタート地点以外の更新不要マスの判別方法について説明する．まず，前提として前マスから注目マスへの移動にかかるコストを 1 とする．そして，事前探索データの注目マスと同地点のマスの値が前マスの到達コストに +1 した値と同値であれば，注目マスは更新不要マスとなり，それ以外であれば，更新必要マスとして探索を行う．

注目マスが更新不要マスの場合，更新不要マスの前マス情報は，同地点マスの事前探索データにある情報をそのまま使用し，その他 3 つの情報は，スタート地点マスと同様に設定する．

注目マスが更新必要マスの場合を以下に説明する．更新必要マスは前マス情報，ヒューリスティックコスト，追加コストを新たに計算する必要がある．

ここで，注目マスと同地点の事前探索データの到達コストを C とする．また，注目マスからみた前マスの到達コストと追加コストをそれぞれ C_p,A_p とする．このとき，注目マスの追加コスト A は

$$A = C_p + A_p + 1 - C \tag{3}$$

という式で求めることができる．式 (3) は，注目マスの到達コストとして想定される値に対して，不足している到達コストを隣のマスと値の比較を行うことで追加コストを求めている．その後，追加コスト A と到達コスト C，ヒューリスティックコストより合計コストを求め，注目マスの情報として設定する．

式 (3) をもとに図 14 における B4 の追加コストを計算すると，

$$A = 3 + 0 + 1 - 2 = 2 \tag{4}$$

となる．その後，B4 の前マスを B5 に設定する．B4,B5,B6 の計算が終了した状態を図 15 に示す．各マスの左上が事前探索データの到達コスト，右上が追加コスト (A)，左下がヒューリスティックコスト，右下が合計コストを示している．

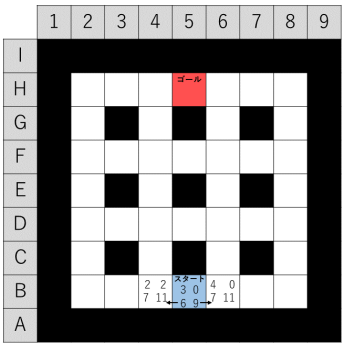

図 15　本手法の初期段階

4.5　マス情報更新の優先順

注目マスの 4 つの情報を設定したら，注目マスに隣接する未探索なマスを抽出し，前回の注目マスを「前マス」と設定して同様の処理を行っていく．その際，未探索マスの更新順番は，合計コストが低いものから優先するものとした．

例えば図 16 の状態の場合，次に情報を更新する候補は B2 と D4 であるが，その際に合計コストが低いマスに隣接するマスを優先して処理を行うため，D4 を先に更新することになる．

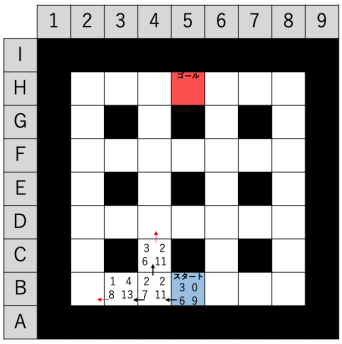

図 16　合計コストによる優先順

D4 の合計コストを算出したのちに，改めて B3 の合計コストと比較を行い，合計コストが低い方のマスの隣接マスで処理を進めていくことになる．これにより，A* と同様な方針で更新するマスを削減できる．

同様の手順でゴール地点までマス情報を更新した様子を図 17 に示す．

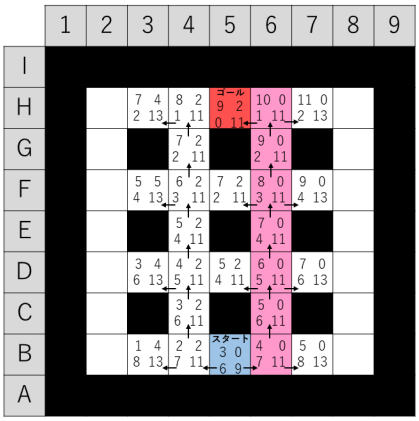

図 17　本手法の探索終了状態

4.6　経路の確定

前節までの処理でゴール地点までのマス情報を更新した後に，ゴールから再帰的に「前セル」を辿り，リストに格納する．これをスタート地点が前セルに出現するまで繰り返す．

スタート地点が出現した時点で，格納リストを反転することで，最短経路情報リストを取得できる．

4.7　本手法のまとめ

A*アルゴリズムと本手法の違いは，A*アルゴリズムでは探索データを 1 つしか持たないのに対して，本手法では，探索データを複数用意しておくことである．本手法では，事前探索データ群から探索する状況に一番近いデータを用いることで，再探索の際にコスト算出対象となるマスを A* よりも大幅に減らし，探索処理時間の削減を図っている．

5　提案手法の検証

この実験では，Fine Kernel ToolKit[13] を用いて，プログラムを作成した．本手法と A*アルゴリズムを用いて複数のマップで経路探索を行い，処理速度を比較した．また，事前探索データ群の作成方針が処理速度へ与える影響を明らかにする為，複数の事前探索データ群間で比較を行った．次に Moving Target D* Lite が適用できる条件下において，本手法，A*アルゴリズム，Moving Target D* Lite の 3 手法の処理速度比較を行った．

5.1　本手法と A*アルゴリズムの比較実験

本手法と A*アルゴリズムを用いて複数のマップで経路探索を行い，処理速度を比較した．まず実験開始時点

でスタート地点とゴール地点を設定し，最短経路を探索する．その後，開始時点とは別のスタート地点とゴール地点を設定し，改めて最短経路探索を行う．開始時点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，二回目のスタート地点とゴール地点を新スタート地点，新ゴール地点と呼ぶものとする．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． このとき，新スタート地点と新ゴール地点はいずれも旧最短経路上ではない箇所に設定されるものとした．

旧最短経路と新最短経路の例をそれぞれ図 18, 図 19 に示す．両図の水色がスタート，赤色がゴール，ピンク色のマスが最短経路を示している．また，図 19 のスタートとゴールは，図 18 の最短経路上には設定されないことになる．

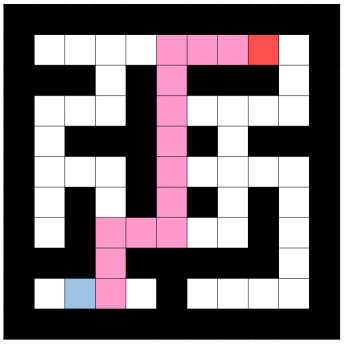

図 18　旧最短経路

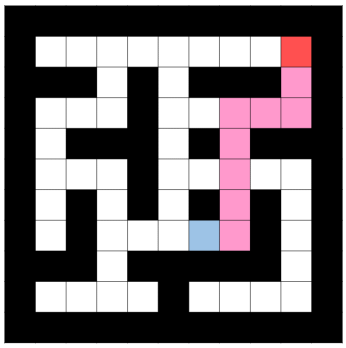

図 19　新最短経路

マップ構造には様々な特徴があるため本実験では，迷路のように入り組んだ構造，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 種類を想定した．迷路のマップを図 20 に，ほぼ一本道のマップを図 21 に，規則的なマップを図 22 に示す．

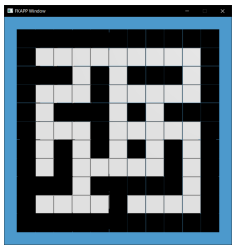

図 20　迷路構造

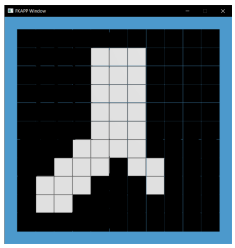

図 21　ほぼ一本道の構造

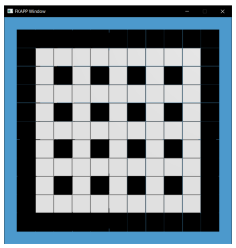

図 22　規則的な構造

図 20〜図 22 の構造を持つ，縦横 5001 × 5001 のサイズのマップを用いて実験を行った．各マップでの計測結果を表 1 に示す．

表 1　本手法と A*アルゴリズムの比較 (ms)

	A*	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

5.2　事前探索データの違いによる比較実験

異なる事前探索データ群を使用した場合の差も検証を行った．

4.1 節で説明した図 10 の対角線上に探索を行った事前探索データ群の他に，対辺上に探索を行った事前探索データ群を用意した．対辺上に探索した事前探索データ群を図 23 に示す．

前節で用いた各マップに対し，両方のデータを用いた探索の処理時間を表 2 に示す．

表 2　本手法の事前探索データ群間の比較 (ms)

	対角線上	対辺上
迷路マップ	6.643	18.462
ほぼ一本道のマップ	1.481	1.588
規則的なマップ	2.924	15.829

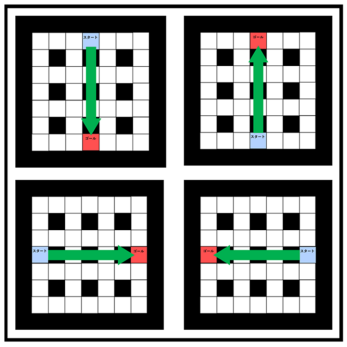

図 23　対辺上に探索を行った事前探索データ群

5.3　本手法と Moving Target D* Lite の比較実験

Moving Target D* Lite が適用できる条件下で A*と Moving Target D* Lite，本手法の処理速度比較を行った．Moving Target D* Lite は，新規スタート地点が最短経路上にある場合に有効な手法であるが，最短経路上にない場合は利用できない．そのため，A*や本手法に比べて使用できる状況下は限られる．しかし，使用条件を満たす状況では，本手法と比べて高速な処理となる可能性もあるため，比較実験を行った．図 18 の状況から，Moving Target D* Lite が適用可能な状況下である，新スタート地点を旧最短経路上に設定した例を図 24 に示す．比較結果を表 3 に示す．本手法の事前探索データ群は対角線上の群を用いた．

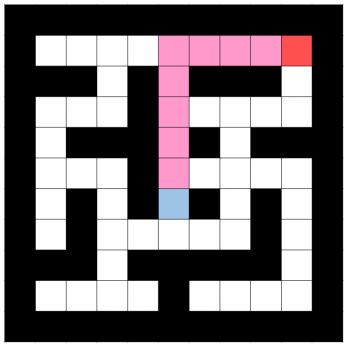

図 24　Moving Target D* Lite が適用可能な状況下での経路探索

表 3　新スタート地点が旧最短経路上を移動した場合 (ms)

	MTD* Lite	A*	本手法
迷路マップ	0.000002	4.886	5.301
ほぼ一本道のマップ	0.002	1.670	1.445
規則的なマップ	0.299	4.469	2.849

5.4　考察

検証結果から，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，新スタート地点を旧最短経路上に設定でき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，新スタート地点を旧最短経路外に設定する場合で，かつ迷路マップ以外で最適であった．

今回は 2 つの事前探索データ群を用意して 3 つのマップでの検証を行ったが，全てのマップにおいて対角線上に探索を行ったものが良い結果となった．対辺上に探索を行った事前探索データ群は対角線上のものに比べ，更新すべきマスの量が増えたことで，処理時間が大幅に増えた．しかし，十字架のようなマップでは対辺上に探索を行った事前探索データ群が有効であることが想像できるため，この結果はマップの形状によって，大きく変化する可能性がある．そのため，それぞれのマップでの事前探索データ群の比較を行う必要があるという課題が挙げられる．この事前探索データ群の事前探索データの種類を増やすことで，より最適な事前探索データを選択することができる可能性があるが，単純に増やしてしまうと，必要となるデータ量が増加してしまうため，必要最低限に抑えなければならない．また，実際のゲームで使用することを想定したとき，単純なマップよりも複雑なマップの形状であることが多いため，今回の迷路マップでの事前探索データ群を最適化する方法を今後考察していく必要がある．

6　まとめ

本研究では，事前探索データを用いた新たな経路探索アルゴリズムを提案し，先行手法である A* アルゴリズムと Moving Target D* Lite との速度比較を行った．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を旧最短経路外に設定した際に，有用であることが判明した．また，今回用意した 3 つのマップでは対角線上に探索を行った事前探索データ群が良い結果となった．

今後の展望として，探索条件に合わせて使用する探索アルゴリズムを変更する処理の考案や最適な事前探索データの追究も必要となる．

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

[7] 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.

[8] 吉田涼真, 阿部雅樹, 渡辺大地. 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. *NICOGRAPH*, No. F-1, pp. 1–8, 2021.

[9] 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.

[10] 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.

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

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

[13] 1997-2020 Fine Kernel Project (fk@gamescience). Fine Kernel ToolKit Top Page. https://gamescience.jp/FK/. 参照: 2021.6.23.

吉田　涼真

2021 年東京工科大学メディア学部卒業．現在，同大学大学院バイオ・情報メディア研究科メディアサイエンス専攻修士課程に在籍．芸術科学会会員．

阿部　雅樹

2008 年東京工科大学メディア学部卒業．2010 年同大学大学院修士課程バイオ・情報メディア研究科メディアサイエンス専攻修了．2016 年より同大学メディア学部実験助手，現在に至る．コンピュータグラフィックスやゲーム制作に関する研究に従事．芸術科学会会員．

渡辺　大地

1994 年慶応義塾大学環境情報学部卒業．1996 年慶応義塾大学政策・メディア研究科修士課程修了．2016 年岩手大学工学研究科デザイン・メディア工学専攻博士後期課程修了．博士 (工学)．1999 年より東京工科大学メディ

ア学部講師．2017 年より同准教授，2020 年より同教授，現在に至る．コンピュータグラフィックスやゲーム制作に関する研究に従事．芸術科学会，情報処理学会，画像電子学会，人工知能学会会員．