芸術科学会論文誌 Vol. 24, No. 3, pp. 8:1 – 8:12 (2025)

マルチエージェント協調における経路探索手法に関する研究

二部孔明 [1)](学生会員)　阿部雅樹 [2)](正会員)　渡辺大地 [3)](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) 3) 東京工科大学メディア学部

A Study on Path Finding in Multi-Agent Cooperation

Komei Nibe[1)]　**Masaki Abe**[2)]　**Taichi Watanabe**[3)]

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

1) g312402466 @ edu.teu.ac.jp 2) abemsk @ edu.teu.ac.jp 3) earth @ gamescience.jp

概要

近年，ゲーム AI に関する技術は著しい発展を遂げており，現在でも盛んに研究が行われている．このゲーム AI を実現する際に重要な技術の 1 つとして，経路探索が挙げられる．経路探索とは，出発地点から目的地点への移動経路を算出することである．ゲーム AI の分野においては，ダイクストラ法や A*アルゴリズムといった経路探索手法が利用されている．これらの手法は最短経路の算出に適しているため，エージェントが特定の目標物を追跡したい場合などに利用することができる．しかし，複数のエージェントが追跡を行うマルチエージェントの環境では，エージェント同士の経路が重複してしまうという問題が発生する可能性がある．この問題を解決し，挟み撃ちのような協調的な追跡を実現できれば，エージェントの動きに高い戦略性を持たせることができる．本研究では，エージェント同士の経路が重複しないように調整し，効率よく追跡ができる経路を算出することを目的とする．本研究の方針は，他のエージェントの経路付近を避けるように経路探索を行うことで，エージェント同士が異なる経路を選択できるようにするものである．また，既存の経路探索手法との比較検証では，提案手法を用いることで経路の重複を回避することができ，挟み撃ちのような動きで効率よく追跡ができることを示した．

Abstract

In recent years, game AI technology has made remarkable progress and is still being actively researched. One of the most important technologies for game AI is path finding. Path finding is the calculation of a travel route from a starting point to a destination point. In the field of game AI, path finding methods such as Dijkstra method and A* algorithm are used. Since these methods are suitable for calculating the shortest path, and can be used when an agent wants to track a specific target. However, in a multi-agent environment, where multiple agents are tracking, there is a possibility of overlapping paths between agents. If this problem can be solved and cooperative tracking, such as a pincer movement, can be realized, agents' movements can be more strategic. The objective of this research is to calculate efficient tracking paths by adjusting the paths of agents so that they do not overlap. The policy of this research is to avoid the vicinity of other agents' paths so that the agents can choose different paths from each other. Comparison with existing path finding methods shows that the proposed method can avoid overlapping paths and can track efficiently in a pincer movement.

1 はじめに

近年，ゲーム AI に関する技術は著しい発展を遂げており, 現在でも盛んに研究が行われている．三宅 [1][2][3] は，ゲーム AI は「キャラクター AI」,「メタ AI」,「ナビゲーション AI」の 3 種類に大きく分類できると述べている．このうち，キャラクター AI は Non Player Character(以下「エージェント」) を実装する際に重要となる技術であり，エージェントの行動制御の役割を担っている．また，メタ AI はゲーム進行の役割を担っており，ナビゲーション AI はキャラクター AI とメタ AI に対して情報を提供する役割を担っている．

キャラクター AI の技術は古くからゲームに用いられており，様々な方法で実装されている．「パックマン」[4] では，各エージェントに異なるアルゴリズムを設定し，固有の性格を持っているかのような動きを実現した．「Killzone」[5] では，マップ上に等間隔にポイントを敷き詰め，各ポイントから情報を抽出して評価することでエージェントの行動を制御した．「クロムハウンズ」[6][7] や「F.E.A.R」[8] では，ゴール指向型プランニングを用いてゲーム状況から多数の目的を設定し，エージェントが適切な方法で目的を達成できるようにした．

キャラクター AI やナビゲーション AI を実現する際に重要な技術の 1 つとして，経路探索が挙げられる．経路探索とは，出発地点から目的地点までの移動経路を算出することであり，幅広い分野で研究が行われている技術である．桑谷ら [9] や佐藤ら [10] は，シューティングゲームにおいて敵の弾を避けるルートを経路探索を用いて求めている．渡辺 [11] や熊谷ら [12] は，波動方程式の波伝搬作用を利用し，追跡行動アルゴリズムを実現する手法を提案した．中宮ら [13] は，移動型センサノードの実用上の問題を考慮し，コストマップを用いた最小コスト経路探索手法を提案した．

複数のエージェントが存在するマルチエージェント環境における経路探索についても，様々な先行研究が存在している．例として,「Vehicle Routing Problem (VRP)」[14] に関する研究が挙げられる．この研究では，エージェントが分担して目的地を効率よく巡回する方法について取り扱っている．柴山ら [15] は，バトルロイヤルゲームにおいて効率よくアイテムを探索する方法として，VRP を応用した手法を提案した．他には,「Multi-Agent Path Finding(MAPF) 問題」に関する研究が挙げられる．MAPF 問題とは，エージェント同士が衝突することなく目的地へ向かう経路を求める問題である．また，MAPF 問題を発展させた,「Multi-Agent Pickup and Delivery(MAPD) 問題」[16] も提案されている．下川ら [17][18] は，MAPD 問題の解法である Token Passing(TP) を改善し，効率的にタスクを割り当てる手法を提案している．

ゲーム AI の分野においては，ダイクストラ法 [19] や A*アルゴリズム [20] といった経路探索手法が多く利用されている．これらの手法は最短経路の算出に適しているため，エージェントが追跡を行う際などに利用することが可能である．例えば，エージェントが 2 体いる場合に，追跡を行うエージェント (以下「追跡エージェント」) と，逃げるエージェント (以下「逃亡エージェント」) の 2 種類に分かれるとする．このとき，ダイクストラ法や A*アルゴリズムを利用すると，追跡エージェントは効率よく逃亡エージェントを追跡することができる．しかし，追跡エージェントが複数体存在する場合は，追跡エージェント同士の経路が重複してしまうという問題が発生する可能性がある．経路の重複は追跡効率の低下につながるため，追跡エージェント同士が異なる経路を選択できるようにすることが求められる．この問題を解決し，挟み撃ちのような協調的な追跡を実現できれば，エージェントの動きに高い戦略性を持たせることができる．

以上を踏まえ，エージェント同士の経路が重複しないように調整し，効率よく追跡ができる経路を算出することを本研究の目的とする．ダイクストラ法，A*アルゴリズム，VRP，MAPF，MAPD 問題などの理論では最短経路を算出するため，経路が重複してしまう可能性がある．そこで，本手法では他のエージェントの経路付近を避けるように経路探索を行い，エージェント同士が異なる経路を選択できるようにする．また，我々は本研究の初期成果を NICOGRAPH2024 にて発表した [21]．これに対し，本手法のゲーム AI への応用可能性を評価するため，既存の経路探索手法との比較検証をさらに充実させた．具体的には，広大なマップでの検証に加え，一方通行マスや移動コストの高いマスを導入し，より実践的なシチュエーションでの検証を行った．

2 提案手法

2.1 経路探索の準備

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する．エージェントが移動可能なマスを通路マス，侵入することができないマスを壁マスとする．図 1 は，エージェントの配置例であり，本章ではこのマップを用いてアルゴリズムの説明を行う．通路マスを白色，壁マスを黒色で表しており，水色の三角形が追跡エージェント，赤色の円形が逃亡エージェントを表している．

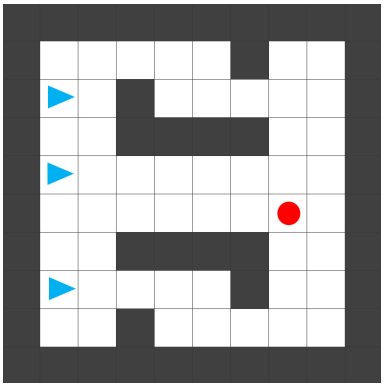


図 1: エージェントの配置例

本手法では追跡エージェントが 1 体ずつ順番に経路探索を行う．追跡エージェントは，逃亡エージェントとの距離が近い順に経路探索を行い，距離の計算にはマンハッタン距離 [22] を用いる．マンハッタン距離とは，2 点の横方向と縦方向のずれを足した数値のことである．マンハッタン距離を用いて距離の計算を行うと，マップ中央に位置している追跡エージェントが最も近いという結果が得られる．そのため，マップ中央に位置している追跡エージェントから経路探索を行う．

また，本章ではエージェントが移動可能な方向は上下左右の 4 方向に制限する．ただし，エージェントが斜めを含めた 8 方向に移動可能な場合は，マンハッタン距離の代わりにチェビシェフ距離を用いて計算を行う．チェビシェフ距離とは，2 点の横方向と縦方向のずれのうち大きい方の数値のことである．

2.2 1 体目の追跡エージェントの経路探索

1 体目の追跡エージェントは，経路探索手法であるダイクストラ法と同様の手順で経路を算出する．ダイクストラ法では，スタート地点からゴール地点までのコストを算出し，コストマップを生成することで最短経路を求めることができる．手順としては，はじめにスタート地点の上下左右に存在する通路マスにコストを設定する．このコストはスタート地点からの移動コストを表しており，1 マス移動するのにかかるコストは 1 である．なお，1 度設定したコストを上書きすることはできず，壁マスにはコストを設定することができない．図 2 は，スタート地点の周りにコストを設定した様子である．スタート地点のマスを青色，ゴール地点のマスを赤色で表している．

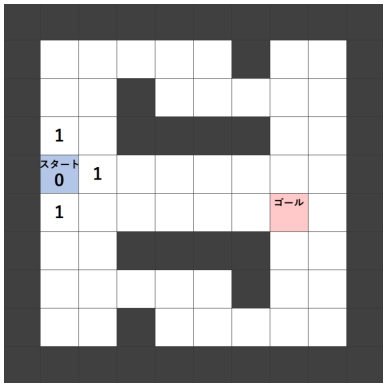


図 2: スタート地点の周りにコストを設定した様子

次に，コスト 1 を設定したマスの周りにコスト 2 を設定し，その次はコスト 2 を設定したマスの周りにコスト 3 を設定する．このように，周囲のマスのコストを算出する作業を繰り返していく．本手法では，ゴール地点のコストを算出した時点で探索を終了するため，全てのマスのコストを算出する必要はない．図 3 は，コストの算出が完了した様子である．

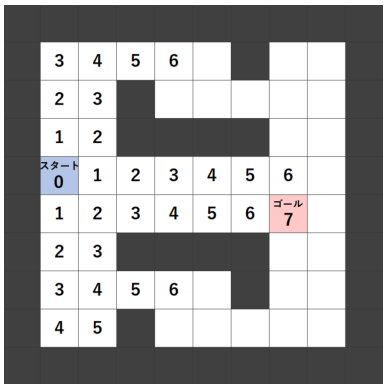


図 3: コスト算出結果

最後に，ゴール地点からコストが 1 ずつ低くなるよ

うにマスを遡っていき，スタート地点にたどり着くまで遡る作業を繰り返す．このとき，スタート地点まで遡るために経由したマスが経路となる．マスを遡る際に同じコストのマスが現れることがあるが，その場合はどちらのマスを選択しても経路を算出することができる．図 4 は，マスを遡ることで経路を算出した様子であり，算出した経路は水色の矢印で表している．

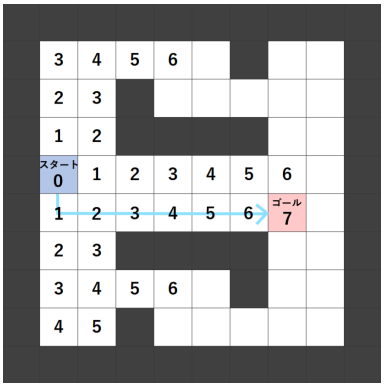


図 4: 経路算出結果

このように，1 体目の追跡エージェントはダイクストラ法のアルゴリズムを利用するため，特別な処理などは行わない．

2.3 2 体目の追跡エージェントの経路探索

2 体目の追跡エージェントは，1 体目の追跡エージェントの経路付近を避けるように経路探索を行う．図 5 は，1 体目の追跡エージェントの経路を黄色で表し，2 体目の追跡エージェントの経路探索の準備をしたものである．

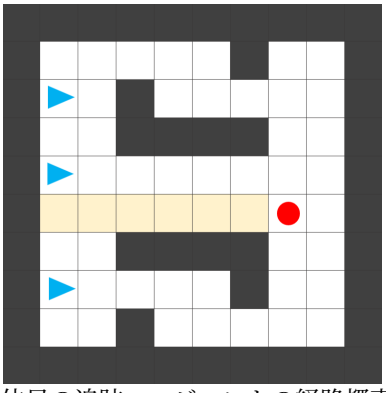


図 5: 2 体目の追跡エージェントの経路探索の準備

2 番目に距離が近いのはマップ下部に位置している追跡エージェントであるため，次はこの追跡エージェントが経路探索を行う．図 6 は，1 体目の追跡エージェントの経路付近を色分けした様子である．1 体目の追跡エージェントの経路は黄色で表しており，経路に隣接するマスは緑色で表している．

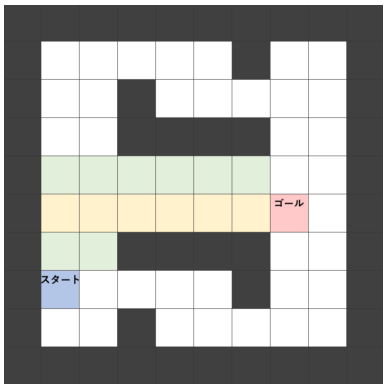


図 6: 1 体目の追跡エージェントの経路付近を色分けした様子

本手法では，黄色のマスと緑色のマスを避けるようにコストを算出し，追跡エージェント同士の経路が重複しないようにする．黄色のマスを避けるだけでも経路の重複は回避できるが，さらに緑色のマスも避けることで，経路同士が隣接しないように調整している．また，経路を算出する際は，ダイクストラ法と同じ手順で経路を算出する．図 7 は，コストを算出した様子であり，算出した経路は水色の矢印で表している．

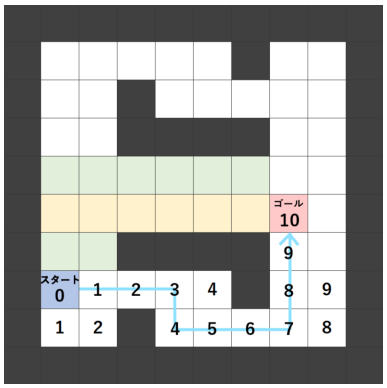


図 7: 2 体目の追跡エージェントの経路算出結果

このように，経路付近を避けるようにコストの算出を行うことで，2 体目の追跡エージェントは最短経路以外の別の経路を算出することができる．

2.4 3 体目の追跡エージェントの経路探索

3 体目の追跡エージェントは，1 体目と 2 体目の追跡エージェントの経路情報を利用して経路探索を行う．図

8 は，1 体目と 2 体目の追跡エージェントの経路を黄色で表し，3 体目の追跡エージェントの経路探索の準備をしたものである．

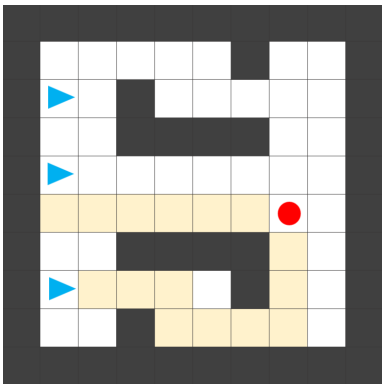


図 8: 3 体目の追跡エージェントの経路探索の準備

3 体目の追跡エージェントが経路探索をする際は，1 体目と 2 体目の追跡エージェントの経路付近を避けるようにコストを算出する．2 体目の場合と同様に，黄色のマスと緑色のマスを避けるようにコストを算出し，他のマスはダイクストラ法と同じ手順でコストを算出する．図 9 は，コストを算出した様子であり，算出した経路は水色の矢印で表している．

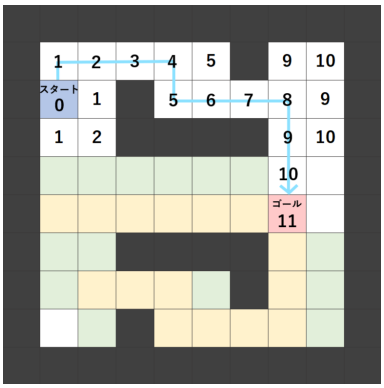


図 9: 3 体目の追跡エージェントの経路算出結果

このように，3 体目の追跡エージェントも，本手法を用いることで最短経路ではない別の経路を算出することができる．本手法は，追跡エージェントの数が増えても同様の手順で経路を算出することが可能であり，追跡エージェントの数が何体でも対応することができる．

2.5 ゴール地点までの経路が算出できない場合

本手法では経路付近を避けるようにコストの算出を行うため，ゴール地点までの経路が算出できない場合がある．図 10 は，2 体目の追跡エージェントの経路探索時に，経路が算出できない場合を表している．1 体目の追跡エージェントの経路は黄色で表しており，経路に隣接するマスは緑色で表している．

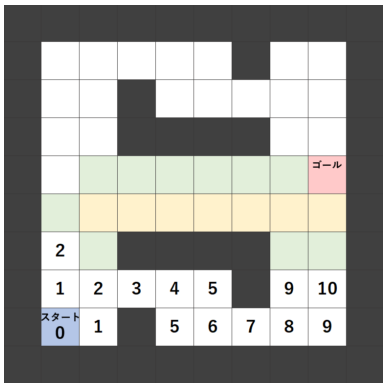


図 10: 経路が算出できない場合

経路を算出できない場合は，緑色で色分けしているマス，すなわち経路に隣接しているマスのコスト算出を行う．図 11 は，緑色のマスのコストを算出し，ゴール地点までのコスト算出を行った様子である．

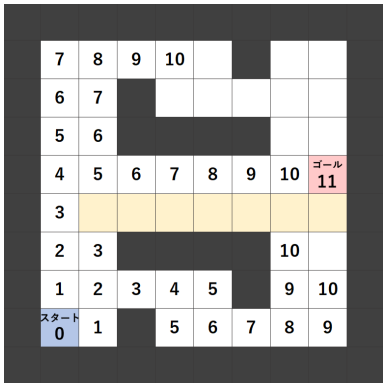


図 11: コスト算出結果

このように，緑色のマスのコストを算出することで探索を再開し，ゴール地点までのコストを算出することができる．この例だと挟み撃ちを行うことはできないが，追跡エージェント同士の経路が重複することは回避している．しかし，エージェントの配置状況によっては，緑色のマスのコストを算出しても経路が見つからない場合がある．その場合は，黄色で色分けしているマス，すなわち追跡エージェントの経路上のマスのコストを算出する．このような手順にすることで，スタートとゴールが非連結でない限り，最終的に 1 つも経路が算出できない

という状況が発生しないようにしている．

2.6 経路算出結果

図 12 はダイクストラ法の経路算出結果で，図 13 は提案手法の経路算出結果である．算出した経路のマスを黄色で表しており，それぞれの追跡エージェントから逃亡エージェントまでの経路を水色の矢印で表している．

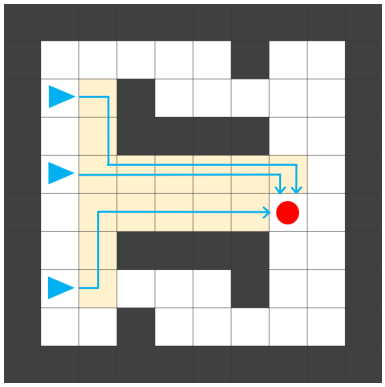


図 12: ダイクストラ法の経路算出結果

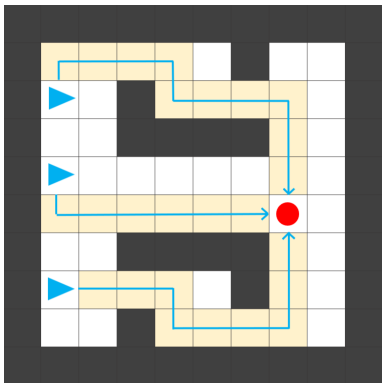


図 13: 提案手法の経路算出結果

ダイクストラ法の場合は追跡エージェント同士の経路が重複しているが，提案手法の場合は追跡エージェント同士が異なる経路を選択することができている．このように，提案手法を用いることで，挟み撃ちのような協調的な動きができる経路を算出することが可能である．また，本章ではエージェントが上下左右の 4 方向に移動可能な場合の例を紹介しているが，斜めを含めた 8 方向に移動可能な場合も同様の手順で経路を算出することが可能である．

3 提案手法の検証

3.1 検証方法について

提案手法の検証を行うために，Unity ゲームエンジン上に本手法を実装した．本検証では，ダイクストラ法，A*アルゴリズム，提案手法の 3 つの経路探索手法を利用し，経路探索の結果を比較する．評価をする項目は，「追跡エージェントの追跡の様子」，「逃亡エージェントに追いつくまでにかかるフレーム数」，「経路探索にかかる処理時間」，「追跡エージェント同士の距離」の 4 点である．「追跡エージェントの追跡の様子」については，検証の様子を表す図と文章を用いて検証結果の説明を行う．それ以外の項目については，表を用いてデータの比較を行う．

検証環境はフレームレートを 60fps に設定し，エージェントの移動速度は 0.05 マス／フレームとする．追跡エージェントおよび逃亡エージェントは 15 フレームごとに経路探索を行う．なお，経路算出時にコストが等しいマスが複数存在し，いずれを選んでも問題がない場合は乱数により移動先を選択する．

本検証では，追跡エージェントが逃亡エージェントに追いついた時点で検証を終了する．この判定のために，追跡エージェントの中心と逃亡エージェントの中心との距離を毎フレーム計算し，その距離が 1.00 マス以下になった場合に追いついたと判定する．距離の算出には，両者間の単純な直線距離を測定する目的から，ユークリッド距離を使用する．また，算出経路によっては追跡エージェント同士が重なることがあるが，本検証においてはエージェント同士の重なりを許容する．

検証ではエージェントの数や配置，マップの地形などを変更し，計 9 種類の検証を実施する．具体的な検証内容は以下の通りである．

- 検証 1: シンプルな地形のマップを用いた検証
- 検証 2: 複雑な地形のマップを用いた検証
- 検証 3: 広大な地形のマップを用いた検証
- 検証 4: 広大で複雑な地形のマップを用いた検証
- 検証 5: 一方通行の場所がある場合の検証
- 検証 6: 部分的に移動コストを増やした場合の検証
- 検証 7: 配置変更で経路重複を回避した場合の検証
- 検証 8: 追跡エージェントを 3 体にした場合の検証
- 検証 9: 斜めを含めた 8 方向に移動可能な場合の検証

各検証においては，エージェントの位置などの初期条件を統一し，経路探索手法のみを変更して比較を行った．

3.2 逃亡エージェントについて

逃亡エージェントの経路算出方法をわかりやすく説明するために，まずは追跡エージェントが 1 体の場合を想定して説明する．逃亡エージェントは，追跡エージェントから逃げるために，追跡エージェントの位置をスタート地点としてコストマップを生成する．このコストマップの生成には，ダイクストラ法を利用する．図 14 は，追跡エージェントの位置をスタート地点として，コストを算出した様子である．水色の三角形が追跡エージェント，赤色の円形が逃亡エージェントを表している．

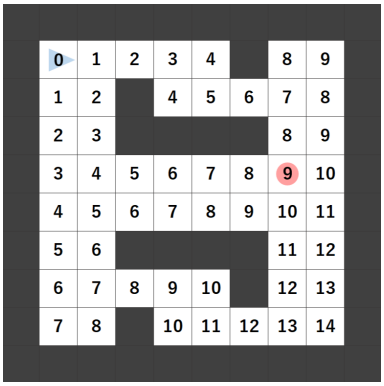


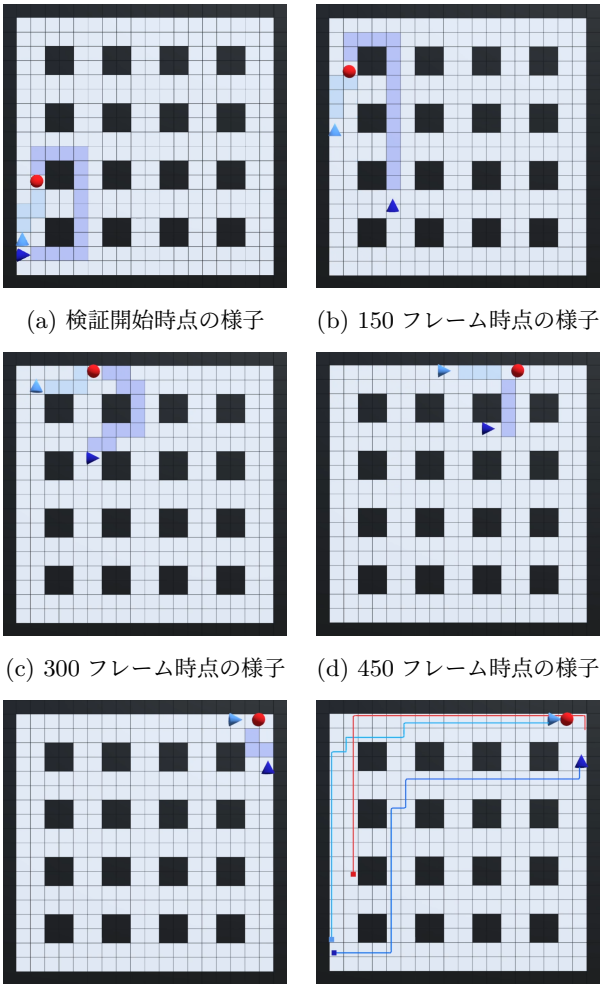
図 14: コスト算出結果

逃亡エージェントは周囲のマスのコストを参照し，最もコストの高いマスへの移動を繰り返す．この例だと，逃亡エージェントの右のマスと下のマスのコストが 10 であり，上下左右のマスの中だと最もコストが高い．そのため，次の移動先は右のマスか下のマスのどちらかになる．このような処理を繰り返すことで，逃亡エージェントは追跡エージェントから離れるように移動する．

追跡エージェントが 2 体以上いる場合は，それぞれの追跡エージェントごとに同様のコストマップを生成する．そして，各マスにおいてそれらすべてのコストの平均値を算出し，逃亡エージェントは平均コストが最も高い周囲のマスへの移動を繰り返す．

3.3 検証結果

検証 1 では 20 × 20 マスのシンプルな地形のマップを用いて検証を行った．図 15 は，提案手法の検証を行っている様子である．図 15a から図 15e では，追跡エージェントが算出した経路上のマスを色分けして表している．図 15f は，追跡エージェントが逃亡エージェントに追いついた時点の様子である．各エージェントの移動軌跡を表示しており，軌跡の開始地点には四角形のマークを付けている．なお，軌跡同士が重なって視認性が低下することを防ぐため，軌跡はエージェントの中心位置からわずかにずらして表示している．



(a) 検証開始時点の様子　(b) 150 フレーム時点の様子

(c) 300 フレーム時点の様子　(d) 450 フレーム時点の様子

(e) 600 フレーム時点の様子　(f) 追いついた時点の様子

図 15: 検証 1 の様子

提案手法を利用した場合は，検証開始から 608 フレーム経過した時点で逃亡エージェントに追いつくことができた．追跡エージェントは最短経路ではない別の経路を算出しており，挟み撃ちのような効率的な追跡を行うことができた．一方，ダイクストラ法と A*アルゴリズムを利用した場合は，検証開始から 2015 フレーム経過した時点で逃亡エージェントに追いつくことができた．この場合は追跡エージェント同士の経路が重複しており，

単一のエージェントによる追跡と変わらない状況になっていた．

検証 1 では詳しく説明を行ったが，以降の検証でも手順は同様であるため，検証 2 以降は複数の図を用いた詳細な説明は省略する．図 16 から図 23 は，検証 2 から検証 9 までの提案手法の様子を表している．また，表 1 では，追跡エージェントが逃亡エージェントに追いつくまでにかかるフレーム数を表している．

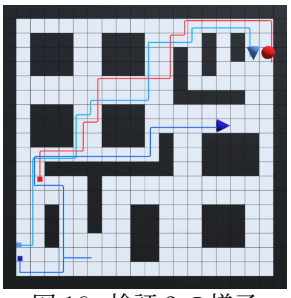


図 16: 検証 2 の様子

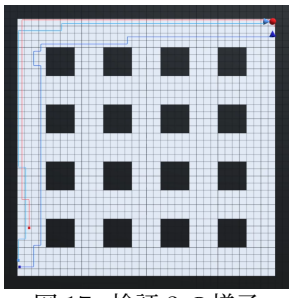


図 17: 検証 3 の様子

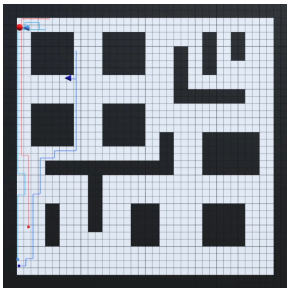


図 18: 検証 4 の様子

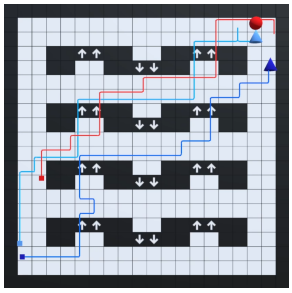


図 19: 検証 5 の様子

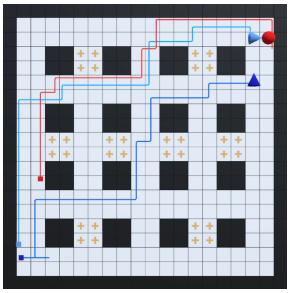


図 20: 検証 6 の様子

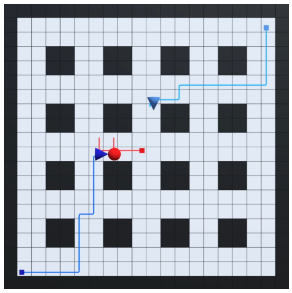


図 21: 検証 7 の様子

検証 2 では，検証 1 よりも複雑な地形のマップを用いて検証を行った．通路の幅が狭い箇所や広い箇所があり，実際のゲームに近い地形を模している．提案手法を用いることで経路の重複を回避し，挟み撃ちのような効率的な追跡ができた．

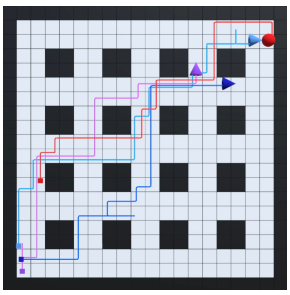


図 22: 検証 8 の様子

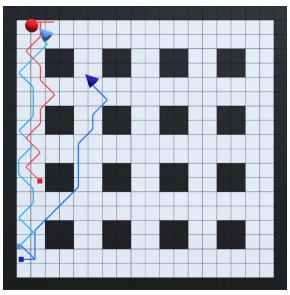


図 23: 検証 9 の様子

表 1: 逃亡エージェントに追いつくまでにかかるフレーム数

	ダイクストラ，A*	提案手法
検証 1	2015	608
検証 2	1460	668
検証 3	10038	1428
検証 4	6342	888
検証 5	6036	649
検証 6	6924	649
検証 7	273	273
検証 8	3388	649
検証 9	1491	383

検証 3 では，40 × 40 マスの広大な地形のマップを用いて検証を行った．これは，広大な地形における経路探索の性能や処理時間を確認するために行ったものである．提案手法では，1 体の追跡エージェントが最大 3 マス幅を使うため，4 マス以上の通路幅があるマップを用意した．提案手法を用いることで経路の重複を回避し，挟み撃ちのような効率的な追跡ができた．

検証 4 では，検証 3 よりも複雑な地形を用いて，40 × 40 マスのマップで検証を行った．これは，広大さと複雑さを兼ね備えた実践的なマップでの性能を評価するために行ったものである．提案手法を用いることで経路の重複を回避し，挟み撃ちのような効率的な追跡ができた．

検証 5 では，矢印が示す方向にしか進めない一方通行マスを用意して検証を行った．これは，ゲームにおける段差や崖などの移動制限を表現するために，一方通行の箇所を用意したものである．提案手法を用いることで経路の重複を回避し，一方通行に対応しながら挟み撃ちのような効率的な追跡ができた．

検証 6 では，黄色の目印がある特定のマスの移動コストを 5 倍に設定して検証を行った．これは，ゲームにおける沼などの移動に制限がある地形を表現するために，移動コストの調整を行ったものである．提案手法を用いることで経路の重複を回避し，コスト増加に対応しながら挟み撃ちのような効率的な追跡ができた．

検証 7 では，追跡エージェントを対角線上に配置し，経路の重複が起こらないようにして検証を行った．この検証では，どの経路探索手法を用いても同じ経路が算出され，逃亡エージェントに追いつくまでの時間は変わらなかった．これは，配置状況が変わったことで追跡エージェント同士の経路が重複しなくなり，提案手法を利用しない場合でも挟み撃ちができたからである．

検証 8 では，追跡エージェントの数を 3 体に増やして検証を行った．2 体の場合と同様に，提案手法を用いることで経路の重複を回避し，挟み撃ちのような効率的な追跡ができた．しかし，マップの広さに限りがあるため，3 体すべてが完全に異なる経路を選ぶのは難しく，一部で経路の重複が見られた．

検証 9 では，斜めを含めた 8 方向に移動可能な場合の検証を行った．近年のゲームでは斜め移動に対応していることが多く，より現実的な条件下での動作を確認するために行ったものである．4 方向移動の場合と同様に，提案手法を用いることで経路の重複を回避し，挟み撃ちのような効率的な追跡ができた．

3.4 経路探索にかかる処理時間

経路探索にかかる処理時間については，経路探索を行うたびに処理時間を測定し，検証終了後にその平均値を算出した．ただし，経路探索手法ごとに 1 回の検証で実行される経路探索の回数が異なるため，公平に比較するためにダイクストラ法と A*アルゴリズムは測定回数を調整した．例えば，検証 1 の提案手法では 41 回の経路探索を行っているため，ダイクストラ法と A*アルゴリズムも最初の 41 回の経路探索のデータを用いて平均値を算出した．

表 2 では，経路探索にかかる処理時間の平均値を経路探索手法ごとに表している．処理時間の単位はマイクロ秒を用いており，小数点第 3 位以下の値は切り捨てている．

検証 3 と検証 4 を除くすべての検証において，最も処理時間が短かったのはダイクストラ法であった．40 × 40 マスのマップを用いた検証 3 および検証 4 では，A*アルゴリズムが最も処理時間が短かった．これは，マップが広大になるほど，A*アルゴリズムの方が効率的に経路探索を行うことができるからである．そのため，提案手法が最も処理時間が短いケースは確認できず，本手法は経路探索にかかる処理時間が比較的長いという結果が得られた．ただし，今回の検証では処理時間の差がマイクロ秒単位であったため，状況によっては処理時間の差が経路探索に影響を与えない場合もある．

表 2: 経路探索にかかる処理時間の平均値 (マイクロ秒)

	ダイクストラ	A*	提案手法
検証 1	37.47	51.74	47.94
検証 2	37.31	49.71	65.66
検証 3	101.55	45.80	121.98
検証 4	131.90	100.07	186.06
検証 5	37.82	49.27	55.52
検証 6	36.09	48.45	58.84
検証 7	65.31	111.08	74.22
検証 8	53.74	59.91	120.12
検証 9	47.73	73.38	68.96

3.5 追跡エージェント同士の距離

追跡エージェント同士の距離については，マンハッタン距離で毎フレーム測定を行い，検証終了後にその平均値を算出した．経路探索にかかる処理時間の場合と同様に，公平に比較するためにダイクストラ法と A*アルゴリズムは測定回数を調整した．また，検証 8 は追跡エージェントの数が 3 体であり，検証 9 はチェビシェフ距離を用いているため，データの測定を行っていない．

表 3 では，追跡エージェント同士の距離の平均値を経路探索手法ごとに表している．1 マス進むのにかかる距離を 1 とし，小数点第 3 位以下の値は切り捨てている．

検証 1 から検証 6 では，提案手法を用いることで追跡エージェント同士の距離が大きくなるという結果が得られた．これは，追跡エージェントが密集せずに分散していることを示している．一方，検証 7 では，どの経路探索手法を用いても追跡エージェント同士の距離の平均値に差は見られなかった．これは，エージェントの初期配置状況を変更したことで，どの経路探索手法を利用しても同じ経路が算出されたからである．

表 3: 追跡エージェント同士の距離の平均値

	ダイクストラ，A*	提案手法
検証 1	1.16	7.25
検証 2	1.02	9.03
検証 3	1.27	4.19
検証 4	1.36	9.06
検証 5	1.03	4.91
検証 6	1.27	6.48
検証 7	20.75	20.75

3.6 評価と分析

検証の内容を踏まえ，本手法に関する情報を整理すると以下のような特徴が挙げられる．まず，最短経路以外の経路を算出することができるため，挟み撃ちのような協調的な追跡を実現し，エージェントの動きに戦略性を持たせることができる．次に，本手法はシンプルで実装が容易であるため，既存のゲームに用いられているような複雑なキャラクター AI を準備せずに，効率的な追跡を行うことができる．さらに，経路探索を行うたびにコストマップを再計算するため，マップの地形に変化が生じた際も対応することができる．加えて，追跡エージェント同士の衝突を避けることができているため，MAPF 問題や MAPD 問題にも応用できる可能性がある．

一方で，本手法にはいくつかの課題も存在している．まず，ダイクストラ法や A*アルゴリズムに比べて，経路算出に時間がかかる傾向がある．そのため，検証 7 のように追跡エージェント同士の経路が重複しない場合は，本手法を用いる利点が少ない．また，挟み撃ちを実現するために，エージェントが大きく遠回りをしてしまう場合がある．他にも，本手法をゲームの敵キャラクター AI に適用した場合，追跡性能が高すぎることでプレイヤーのゲーム体験を損なう可能性がある．

4 まとめ

本研究では，マルチエージェントの環境においてエージェント同士の経路が重複してしまうという問題に着目した．そこで，他のエージェントの経路付近を避けるように経路探索を行うことで，エージェント同士が異なる経路を選択できる手法を提案した．既存の経路探索手法との比較検証では，提案手法を用いることで経路の重複を回避することができ，挟み撃ちのような動きで効率よく追跡ができることを示した．また，本手法はエージェントの経路情報があれば容易に実装することができ，多くの状況で利用可能である．そのため，本手法はゲーム AI などに適用することができ，挟み撃ちのような戦略的な動きが求められる場面で有用である．

しかし，本手法は経路探索にかかる処理時間が長い傾向があるため，エージェント同士の経路が重複しない場合は本手法を用いる利点が少ない．そこで，処理時間を削減できるように本手法を改良することが今後の課題として挙げられる．また，高低差がある 3 次元のマップでの検証や，非タイルマップ環境での検証など，さらなる検証を行うことで本手法の有用性を確認していきたい．

参考文献

[1] 三宅陽一郎. ゲーム AI 技術入門. 技術評論社, 2019.

[2] 三宅陽一郎. ディジタルゲームにおける人工知能技術の応用. 人工知能学会論文誌, Vol. 23, No. 1, pp. 44–51, 2008.

[3] 三宅陽一郎. ディジタルゲームにおける人工知能技術の応用の現在. 人工知能学会論文誌, Vol. 30, No. 1, pp. 45–64, 2015.

[4] 岩谷徹, 三宅陽一郎, 高橋ミレイ. ゲーム AI の原点『パックマン』はいかにして生み出されたのか？：岩谷 徹インタビュー. 人工知能学会論文誌, Vol. 34, No. 1, pp. 86–99, 2019.

[5] 三宅陽一郎. Killzone における NPC の動的な制御. https://www.slideshare.net/youichiromiyake/killzonenpc. 参照: 2024-8-8.

[6] 三宅陽一郎. クロムハウンズにおける人工知能開発から見るゲーム AI の展望. https://www.slideshare.net/youichiromiyake/ai-cedec2006. 参照: 2024-6-23.

[7] 三宅陽一郎. Chrome Hounds におけるチーム AI. https://www.slideshare.net/youichiromiyake/y-miayke-igdaaiseminar3rd2007512. 参照: 2024-6-23.

[8] 三宅陽一郎. F.E.A.R におけるゴール指向プランニング. https://www.slideshare.net/youichiromiyake/y-miyake-igdaaiseminar2nd2007210.

参照: 2024-6-23.

[9] 桑谷拓哉, 橋本剛. 熟練プレイヤーレベルを目指す弾幕シューティング AI の開発. 情報科学技術フォーラム, Vol. 12, No. 2, pp. 383–384, 2013.

[10] 佐藤直之, Sila Temsiririrkkul, Luong Huu Phuc, 池田心. Inuence 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 AI プレイヤ. 情報処理学会ゲームプログラミングワークショップ 2016 論文集, pp. 57–64, 2016.

[11] 渡辺大地. 波動方程式による波伝播作用を利用した追跡行動アルゴリズム. 芸術科学会論文誌, Vol. 20, No. 1, pp. 1–9, 2021.

[12] 熊谷樹, 阿部雅樹, 渡辺大地. 3 次元空間における波動方程式による追跡行動アルゴリズムに関する研究. 情報処理学会研究報告, Vol. 2024-DCC-36, No. 14, pp. 1–6, 2024.

[13] 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.

[14] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2002.

[15] 柴山叶, 阿部雅樹, 渡辺大地. バトルロイヤルゲームのアイテム探索に適したマルチエージェント経路探索の定式化. 芸術科学会論文誌, Vol. 22, No. 3, pp. 6:1–6:13, 2023.

[16] Hang Ma, Jiaoyang Li, T. K. Satish Kumar, and Sven Koenig. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, pp. 837–845, 2017.

[17] 下川真典, 松井俊浩. MAPD 問題における滞在の予約と移動予測時間を用いるタスク割り当て. 情報科学技術フォーラム, Vol. 19, No. 2, pp. 255–262, 2020.

[18] 下川真典, 松井俊浩. MAPD の解法 TP における移動範囲の制限の緩和と集荷時間の推定に基づくタスク割り当ての改善. 人工知能学会論文誌, Vol. 37, No. 3, pp. A–L84 1–13, 2022.

[19] E.W.Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, Vol. 1, pp. 269–271, 1959.

[20] 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.

[21] 二部孔明, 阿部雅樹, 渡辺大地. マルチエージェント協調における経路探索手法に関する研究. *NICOGRAPH2024*, No. F-4, pp. 1–8, 2024.

[22] Elena Deza and Michel Marie Deza. *Encyclopedia of Distance*. Springer, 2009.

二部 孔明

2024 年東京工科大学メディア学部卒業. 同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入学. 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 年東京工科大学大学院バイオ・情報メディア研究科修士課程修了. 2023 年東京工科大学大学院後期博士課程修了. 博士 (メディアサイエンス). 2016 年より 2024 年まで東京工科大学メディア学部実験助手. 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.

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．芸術科学会においては，NICOGRAPH プログラム委員長，DiVA 編集委員長，論文委員長，副会長などを歴任し，2024 年より芸術科学会会長．