修士論文　　2022 年度

NPC の行動の最善手探索に関する研究

東京工科大学大学院

バイオ・情報メディア研究科

メディアサイエンス専攻

吉田 涼真

修士論文　　2022 年度

NPC の行動の最善手探索に関する研究

指導教員　渡辺 大地 教授

東京工科大学大学院

バイオ・情報メディア研究科

メディアサイエンス専攻

吉田 涼真

論 文 の 要 旨

論文題目	NPC の行動の最善手探索に関する研究
執筆者氏名	吉田 涼真
指導教員	渡辺 大地 教授
キーワード	ゲーム AI、ゴールベース AI、NPC、 最善手探索、経路探索

[**要旨**]

NPC(Non-Player Character) の行動は AI によって管理されており、その AI をキャラクター AI と呼ぶ。キャラクター AI には複数の手法が存在し、そのうちの一つの手法として、ゴールベース AI がある。ゴールベース AI は、その他のベース AI が局所的な判断による行動決定を行うのに対し、長期的な目標を定め、その目標を達成するための行動手順を求める手法である。ゴールベース AI には、2 つの問題点がある。第一に、処理負荷が高く、大まかな行動でしか行動決定ができないことである。第二に、既存のゴールベース AI では、複数の制約条件を考慮した行動決定ができないことである。

本研究では、制約付き最短経路問題に基づき、特定の条件を満たし、目的地点に到達するという長期的な目標の達成と、最も良いスコアを出す行動決定を選択する最善手探索手法について研究する。本手法では、マップの各地点における行動を経路探索におけるノードとして扱うことで、経路探索と同様の手順で行動決定をしていく。 本稿では、あるノードを 1 度のみ通過できる場合を通過制限あり探索、複数回通過できる場合を通過制限なし探索としてそれぞれ手法の提案を行った。どちらの手法でも、長期的目標である目的地点に到達することを達成し、複数の制約条件を満たす行動経路を算出することができた。通過制限あり探索では、処理を高速に行うことができ、探索状況を変更しても処理時間に差が見られなかった。グラフに制限があるため、使用状況は限定的だが、ゲームでの使用を想定しても高速な処理が期待できることがわかった。また、通過制限なし探索ではグラフに制限がなく、2 度以上同じ地点を通過する場合でも行動経路を算出することが可能である。ゲームでの使用を想定すると、処理速度が遅いため高速化が必要であるが、グラフに制限なく使用できる点からゲーム全般への応用が期待できる。

A b s t r a c t

Title	A study of the search for the best move for NPC behavior
Author	Ryoma Yoshida
Advisor	Taichi Watanabe
Key Words	GameAI, GoalBaseAI, Non-Player Character, Search for the best move, Path Finding

[**summary**]

The actions of NPC (Non-Player Character) are managed by AI, which is called Character AI. There are multiple methods of character AI, one of which is goal-based AI. Goal-based AI is a method in which other base AIs make action decisions based on local judgments, whereas it is a method that defines a long-term goal and seeks action steps to achieve that goal. Goal-based AI has two problems. First, it has a high processing load and can only make action decisions in broad terms. Second, existing goal-based AI cannot make action decisions considering multiple constraints.

Based on the constrained shortest path problem, this study investigates a best move search method to select the best scoring action decision that satisfies certain conditions and achieves the long-term goal of reaching the destination point and also gives the best score. In this method, the actions at each point in the map are treated as nodes in the pathway search, and the action decisions are made in the same procedure as in the pathway search. In this paper, we propose a method for with-passage-restriction search when a node can be passed only once, and a method for no-passage-restriction search when a node can be passed multiple times. Both methods achieved the long-term goal of reaching the destination point and calculated a course of action that satisfies multiple constraints. In the with-passage-restriction search, processing was fast, and there was no difference in processing time when the search situation was changed. Although usage scenarios are limited due to the limitations of the graph, we found that high-speed processing can be expected even for use in games. In addition, in the no-passage-restriction search, there is no restriction on the graph, and even if the same point is passed more than once it is possible to calculate the action path. Assuming use in games, the processing speed is slow, so faster processing is necessary, but the fact that it can be used without any restrictions on graphs means that it can be expected to be applied to games in general.

目 次

図 目 次

第 1 章

はじめに

1.1 研究背景と目的

近年のゲームでは、様々な分野の技術が用いられ、特にゲーム AI[1] が重視されている。ゲーム AI は、古くから研究がなされており、本格的に利用されるようになった著名なゲームとして、「がんばれ森川君 2 号」[2] が挙げられる。現在でも盛んに研究が行われており、ゲーム AI は様々な分野 [3] に発展して研究が進められている。ゲーム AI の一つであるキャラクター AI[4] は、Non-Player Character(以下「NPC」) の動作を管理する役目を持ち、NPC における行動決定の中枢を担っている。本稿における NPC とは、プレイヤーが操作をしない、仲間や敵などのゲーム上に登場するキャラクターのことを指している。「FINAL FANTASY XV」[5] や「Grand Theft Auto Online」[6] を代表に多くのゲームには、NPC が存在している。NPC は、ゲーム上でそれぞれの役目を果たすためにキャラクター AI によって決められた行動をしている。NPC の行動の種類は多岐に渡り、移動や攻撃、回復などが主な行動として挙げられる。キャラクター AI には、いくつもの手法があり、ゲームの種類や適用場面に応じて手法を変更していく必要があるため、これまで多くの研究 [7] がなされている。例えば、森ら [8] は、機械学習を用いることで、オープンワールド環境下において、NPC が環境に応じた行動を自動的に選択できるようなキャラクター AI の提案を行った。Meili ら [9] は、遺伝的アルゴリズムと BP ニューラルネットワークを用いて、少ないデータから NPC の行動を最適化する手法を提案した。Boeda ら [10] は、ゲームプレイに応じた、NPC の成長を実現する手法を提案した。Inworld AI[11] は、NPC にも “リアリティ” を与えるための AI 技術について研究をするために、日本からは NTT ドコモから資金調達をした。Inworld AI のプラットフォームを利用することで、既製のアバターに対して声や言語をはじめとするコミュニケーションを理解する機能 [12] を実装できる。また、ディープラーニングを応用した AI キャラクター [13] の研究もあり、これからのゲームやメタバースに大きな影響を与える技術である。他にも、行動ルールの自動生成に関する研究 [14] や模倣学習を利用した成長する格闘ゲームキャラクタに関する研究 [15]、強化学習を用いた戦術獲得に関する研究 [16] な

どキャラクター AI や NPC に関する研究は機械学習 [17] などをはじめとする様々な技術を組み合わせて行われている。

一般的なゲームにおける NPC の行動は、上記で紹介した技術のように規則的または、プレイヤーキャラクターの行動に応じて選択される。しかし、一部のゲームにおいては、プレイヤーキャラクターの介入なく、NPC の行動が選択される。「The Sims」は、プレイヤーキャラクターが存在せず、プレイヤーはシムと呼ばれるキャラクターを作成し、シムの行動をシミュレーションするゲームとなっている。シムは、欲求や願望と呼ばれる評価値とプレイヤーからの一時的な行動の指示によって行動を選択する。そのため、状況に合わせた多様な行動の選択に適宜対応する AI が実装されている。また、オープンワールドゲームとして有名な「Skyrim」では、プレイヤーキャラクターとは関わりのない NPC も自立的に行動を選択している。オープンワールドゲームの多くは、NPC の行動に規則性が見られる。規則性が見られる主な要因として、管理の容易さと処理削減が考えられる。それに対し、Skyrim における NPC は、主要な NPC 以外にも独自の行動が見られ、プレイヤーの行動履歴に合わせて行動やクエストを変化させる「Radiant AI」,「Radiant Stories 」[18] が取り入れられている。しかし、Skyrim では、NPC に自由な行動をさせた結果、プレイヤーが見ていないところで NPC が倒されてしまい、クエストを完遂できないといった問題も発生している。

キャラクター AI には複数の手法が存在し、そのうちの一つがゴールベース AI[19] である。ゴールベース AI とは、長期的な目標を定めて、その目標を達成するための手順を求めることができる手法である。そのため、目標達成のために必要となる行動を選択することができる。ゴールベース AI は、上記の特徴から NPC の移動経路の算出や行動決定に用いられる。近年、目標達成に必要となる手順を求めることができるゴールベース AI が注目されている。ゴールベース AI は、キャラクター AI におけるその他のベース AI の手法と大きな違いがある。それは、目標達成の確実性である。ゴールベース AI は、目標を達成するための行動に重点を置いており、長期的な判断によって行動を決定していく。ゴールベース AI は、目標を達成することが最優先事項とな

る。これに対し、その他のベース AI は、局所的な判断によって、行動を決定してくため、目標達成よりもその他の優先事項に従って行動決定をする。例えば、敵が近くにいる場合は、回避するなどの優先事項によって、目標の達成ができなくなる。そのため、ゴールベース AI は、目標を達成するという点において秀でている。図 1.1 にそれぞれのベース AI を比較したイメージ図を示す。

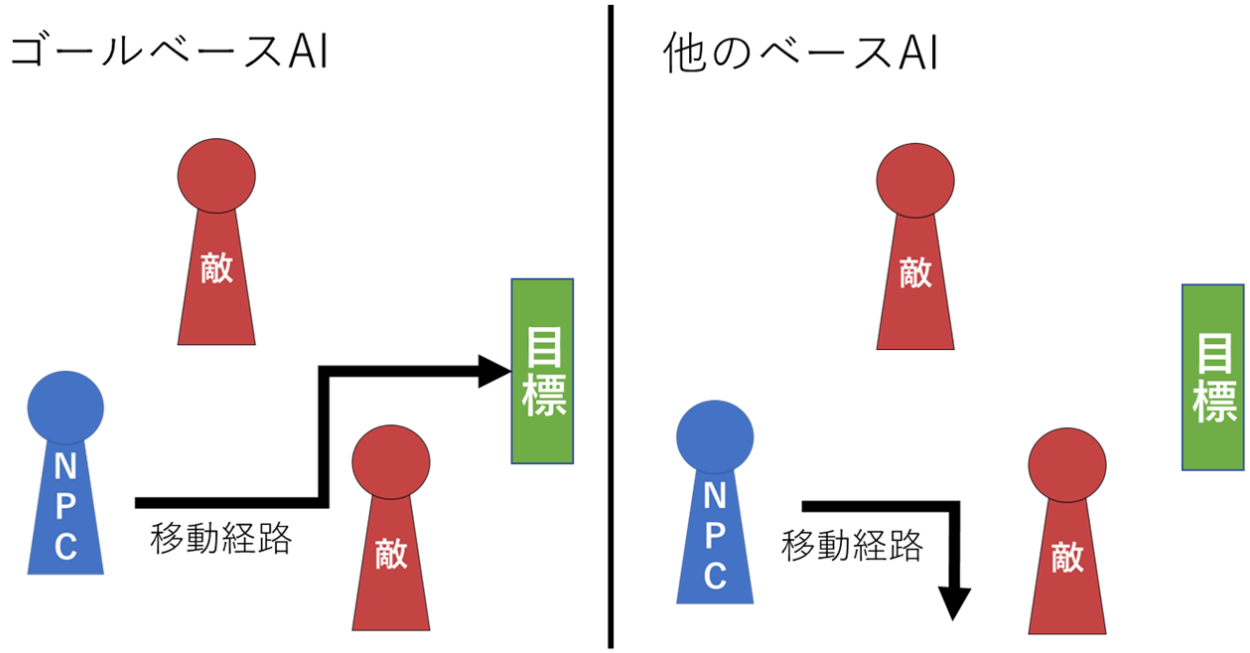

図 1.1　ゴールベース AI とその他のベース AI を比較したイメージ図.

ゴールベース AI に類する 2 つの手法について、それぞれ使用されているゲームと簡単な説明を述べる。

第一に、階層型ゴール指向プランニングである。この手法が用いられた著名な作品として、クロムハウンズ [20] が挙げられる。階層型ゴール指向プランニングは、最終目標であるゴールを小さなゴールに階層化していき、階層化された最小のゴールから徐々に実行することで、最終目標のゴールへと到達する手法である。階層型ゴール指向プランニングのイメージ図を図 1.2 に示す。

第二に、ゴール指向アクションプランニング (以下「GOAP」) である。この手法が用いられた著名な作品として、F.E.A.R.[21] が挙げられる。GOAP は、プランニング時の初期状態から目標

へ到達するために必要となる行動を行動のブロック単位で連鎖的に選択することで目標達成をする手法である。ゴール指向アクションプランニングのイメージ図を図 1.3 に示す。

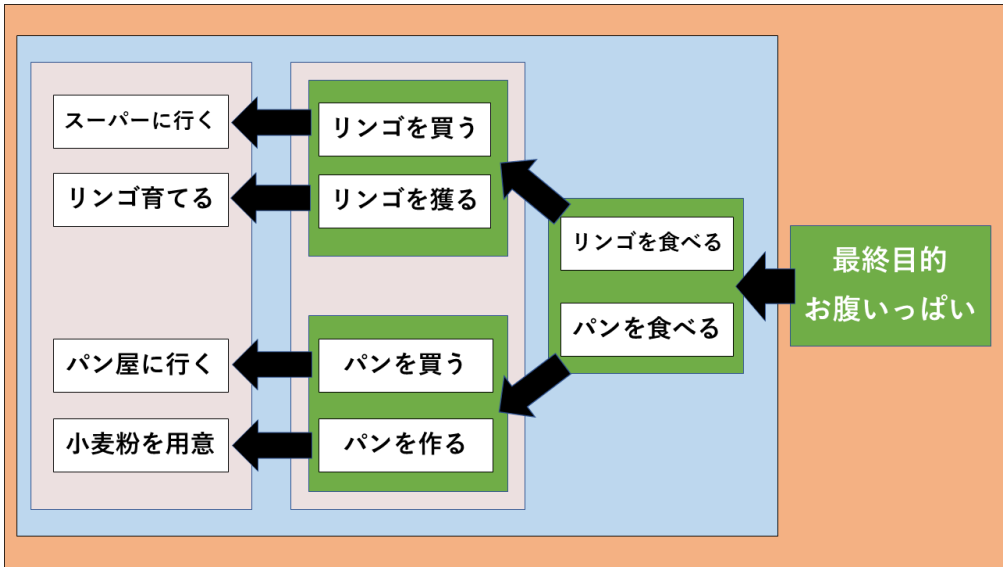

図 1.2　階層型ゴール指向プランニングのイメージ図.

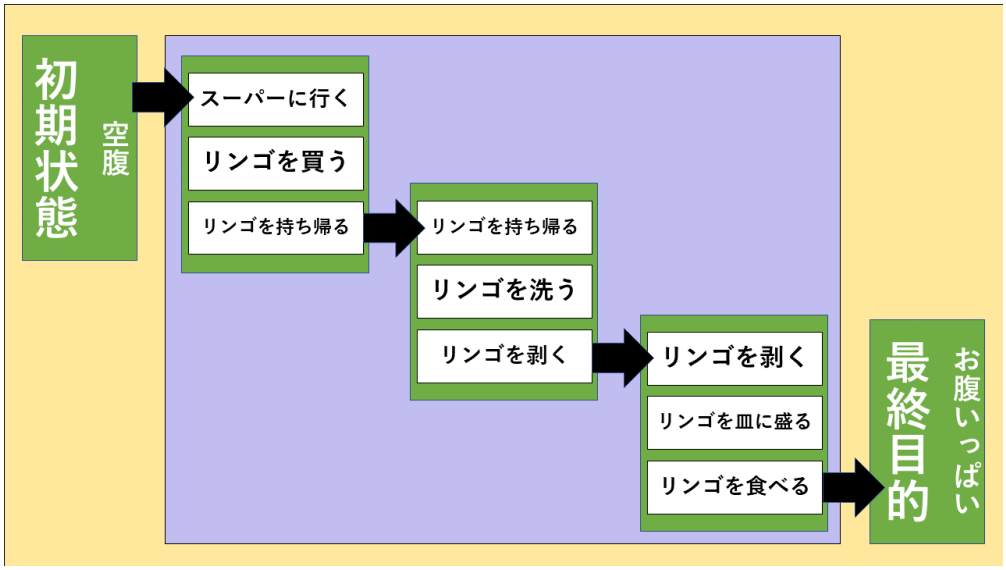

図 1.3　ゴール指向アクションプランニングのイメージ図.

これら 2 つの手法は、目標を達成する行動の算出という点において、非常に有効な手法となる。しかし、これらの手法をゲームで用いる際に起こる大きな問題が 2 つある。

第一に、処理負荷が高いため、大まかな行動でしか行動を管理できないことである。大まかな

行動とは、図 1.2 や図 1.3 に示すようなものである。ゲームでは、

- NPC をはじめとする様々なゲームオブジェクトの生成・削除
- ゲームオブジェクトが移動することで、ゲームマップ上の道の解放・封鎖

が主な原因となり、移動などの行動できる範囲が変化していく。本稿では、上記の 2 つの事象のことを「環境の動的な変化」と呼称する。ゲームでは、こうした環境の動的な変化が高頻度で発生するため、その度にゲーム上で様々な処理が行われる。その処理の 1 つに、NPC の行動探索が含まれる。そのため、ゲーム上で環境の動的な変化が発生するたびに、行動の再探索が必要となる。ゴールベース AI は、長期的な行動を算出するために、探索処理が大幅にかかるという欠点がある。そのため、環境の動的な変化が起きるたびに探索を行うことはゲーム全体の処理に大きな影響を及ぼす。この問題に対して、Restuadi ら [22] は、GOAP を用いてリアルタイムに行動計画を行う手法について提案をした。この研究は、それまでの GOAP とは違い、パラメータを設定することで、リアルタイムでの処理を目指した手法である。しかし、プランニングをするために、パラメータに関連するすべての行動の評価値を設定することが必要があり、それぞれの評価値がわからない場合や、評価ができない場合は利用することができないという問題がある。また、古川ら [23] は、擬似ラプラシアンを用いた盛り土関数を使い、ユーティリティベース AI のゴールベースの特性である推移律が失われず、最終目標を達成する手法の提案を行った。この手法は、移動を重視した手法となっているため、移動以外の行動決定に関しては言及されていない。Restuadi らや古川らの研究のように、処理負荷を軽減するという研究は多く行われている。しかし、行動を細分化して管理するという研究は散見されない。大まかな行動で管理することは処理削減につながるが、行動の不自然さや規則的な行動が発生する。近年のゲームでは、グラフィックをはじめとする性能の向上により、NPC の行動の不自然さが以前よりも目立つようになってきている。そのため、今後のゲームでは、細分化された行動の算出が必要となる。

第二に、複数の条件を満たす行動を算出できないことである。図 1.2 や図 1.3 で示すように、

ゴールベース AI は、目標を達成するためにどのような行動をするかを導くことができる。しかし、ゴールベース AI は、特定の条件を考慮した行動を算出することはできない。例えば、長期的な目標をゴール地点に到達することとした場合、ゴール地点に到達する経路を算出することはできるが、敵に倒されずに多くの敵を倒すなどの複数の条件を考慮した行動を算出することはできない。

複数の制約条件を満たす行動を算出できない原因に、大まかな行動でしか行動決定ができないことが挙げられる。そこで、第一に行動を細分化する方法を考案する。既存のゲームでは、NPC の行動決定と移動経路の算出は、別々に処理を行っている。行動を決定したのちに、移動経路を導き出すという流れになっているため、処理負荷がかかると考えた。そこで、行動決定と移動経路の算出を合わせて処理することで、処理負荷の削減と行動の細分化を図る。第二に、細分化した行動に対して制約付き最短経路問題 [24][25] を適用することで、複数の制約条件を満たす行動を算出することができると考えた。制約付き最短経路問題とは、複数の制約条件を満たす経路を算出する経路探索手法である。制約条件とは、例えば、制限時間やアイテムの回収数などである。この手法は、経路路探索の際に、各地点で条件を満たす経路を記録することで、複数の制約条件がある経路探索の問題に対応する手法である。大まかな行動では、算出できないような制約条件を満たす行動も細分化された行動に適用することで、算出が可能になると判断した。

そこで本研究では、ゲーム上のマップにおける各地点での行動を経路探索におけるノードとして扱い、制約付き最短経路問題に基づいて、複数の制約条件を考慮した行動経路の探索手法を提案する。本手法では、行動決定と移動経路の算出を合わせて行い、経路探索と同様の手順で行動経路を導く手法となる。本手法は、最適な行動を導くのではなく、行動の探索をする際にこれまでの手法ではできなかった、行動決定の細分化と複数の条件を考慮した行動決定ができるという点に重点をおいた研究となっている。

1.2　論文構成

本稿は全 4 章にて構成する。2 章では本手法について述べ、3 章では本研究の提案手法について検証した結果を述べる。そして、4 章ではまとめを述べる。

第 2 章

提案手法

2.1 グラフの定義について

提案手法の説明の前に、本手法で経路算出を行う際に用いるグラフについて説明する。グラフに関する定義・表現方法は、グラフ理論について記述された Wilson[26] の著書に従う。グラフ G は、頂点集合と呼ばれる有限集合 V と辺集合と呼ばれる有限集合 E によって構成され、(V, E) と表せる。頂点集合の V は、グラフ G のノードまたは地点と呼び、辺集合 E は、グラフ G のエッジと呼ぶ。

a から f のノードからなるノードの集合 V とそれらのノード同士をつなぐエッジの集合 E からなるグラフ G を図 2.1 に示す。経路を算出する際に、必ずグラフ G を用いるわけではなく、ノード間をエッジを通り遷移していくというイメージである。後述する行動ノードとの区別をするため、今後 a から f のノードを地点と呼ぶこととする。本稿では、有向グラフを用いて手法の説明を行う。有向グラフとは、図 2.1 に示すように、向きを持つエッジによって構成されるグラフのことであり、エッジの向き、つまり矢印の向きに対してのみ移動ができるものとする。図 2.1 に示した a 地点から b 地点におけるエッジは、a 地点から b 地点に対して矢印が向いているため、a 地点から b 地点に対しては移動できる。しかし、b 地点から a 地点に対しては移動ができない。

本稿では、「移動」と「遷移」の用語を使い分けしている。本研究では、NPC がある地点から別の地点へ移動するという行動としての「移動」と行動を選択するためにノード間を移るというノード上での「移動」があるためである。そこで、NPC が地点間を移る場合に「移動」、NPC が行動ノードを選択する場合に「遷移」を用いる。

2.2 提案手法の方針

提案手法では、以下の 2 つの方針により複数の条件を考慮した行動を算出する。

1. 行動をノードとして、経路探索と同様に算出する
2. 制約付き最短経路問題に基づき、条件を考慮した経路を求める

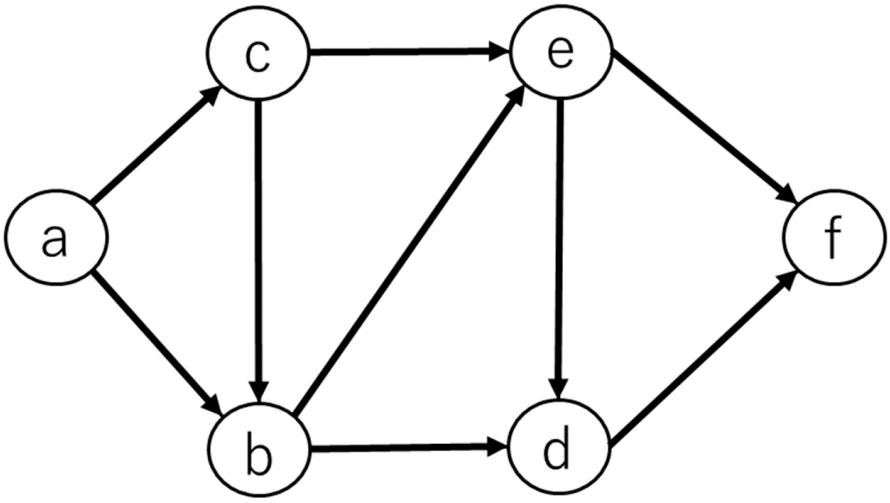

図 2.1　本手法で用いるグラフのイメージ図.

それぞれの方針について説明する。

第一の「行動をノードとして経路探索と同様に算出する」という方針は、既存のゴールベース AI が A*法 [27] を用いて行動決定を行うことができるということに着目し、本手法にも取り入れたものである。A*法は経路探索手法として有名な手法であり、経路探索もゴールベース AI の一部として考えることができる。また、経路探索手法のダイクストラ法 [28] もゴールに到達する経路を求める際に、ゴールを基準として探索を行う点がゴールベース AI と類似している。経路探索手法を用いて行動を算出するために、行動遷移をグラフ化する必要がある。行動遷移とは、初期状態からの行動の分岐を表すものである。また、行動遷移の中で選択された行動を初期状態から結んだものを「行動経路」とする。図 2.2 に示すのは、行動遷移と行動経路のイメージ図である。図 2.2 の初期状態から行動の分岐が起きているものが行動遷移を表し、赤色のエッジで示されているものが、行動経路である。行動の分岐をそれぞれ各地点での行動としてノード化し、それらをまとめたものと「行動グラフ」とする。行動グラフを用いて経路探索を行うことで、行動経路の算出を行う。このようにして、NPC の各地点における行動を経路探索におけるノードに置き換えることで、経路探索と同様に行動の経路を算出することができると考えた。

第二の「制約付き最短経路問題に基づき、条件を考慮した経路求める」という方針は、ゲーム

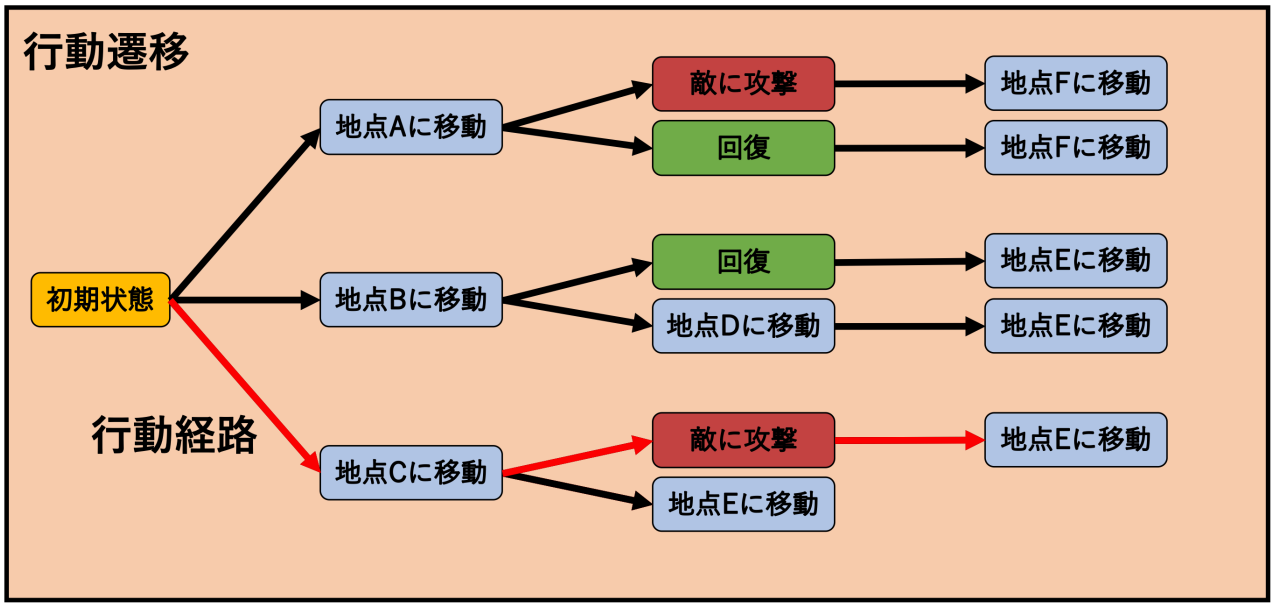

図 2.2　行動遷移と行動経路のイメージ図.

内の制約条件に対して、制約付き最短経路問題と第一の方針と組み合わせることで、条件を考慮した行動経路を算出できるのではないかと考案し、取り入れた。制約付き最短経路問題では、スタート地点からゴール地点に至るまでのエッジに対して、エッジに付与された異なる指標の重みのそれぞれの総和を求め、一方の総和がある一定の範囲内で、もう一方の総和が最大もしくは最小の経路を算出する。これは、ゲームにおける制限時間や攻撃を受けることによるダメージなどの限られた条件のもと、スコアを最も高くする行動の算出に用いることができると考えた。制約付き最短経路問題では、各頂点において条件を満たす経路を記録しておくことで、経路を算出する。

本研究を適用する NPC は、スタート地点からゴール地点へ移動することを長期的な目標とする。その際、複数の条件を考慮した経路を各地点に記録しておき、探索が終了した時点で、ゴール地点から経路を遡ることで、行動を決定していく。

2.3　行動ノードについて

本手法では、各地点での行動を経路探索におけるノードとして扱い、複数のノードを同地点に設定する。行動グラフでの各地点における行動のノードを「行動ノード」と呼称する。各地点における行動ノードは、同地点のそれぞれのノードとエッジで繋がっている。図 2.1 における b 地点と d 地点におけるそれぞれの行動ノードとエッジのイメージ図を図 2.3 に示す。図 2.3 のグラフは、2.2 節で紹介した行動グラフとなる。

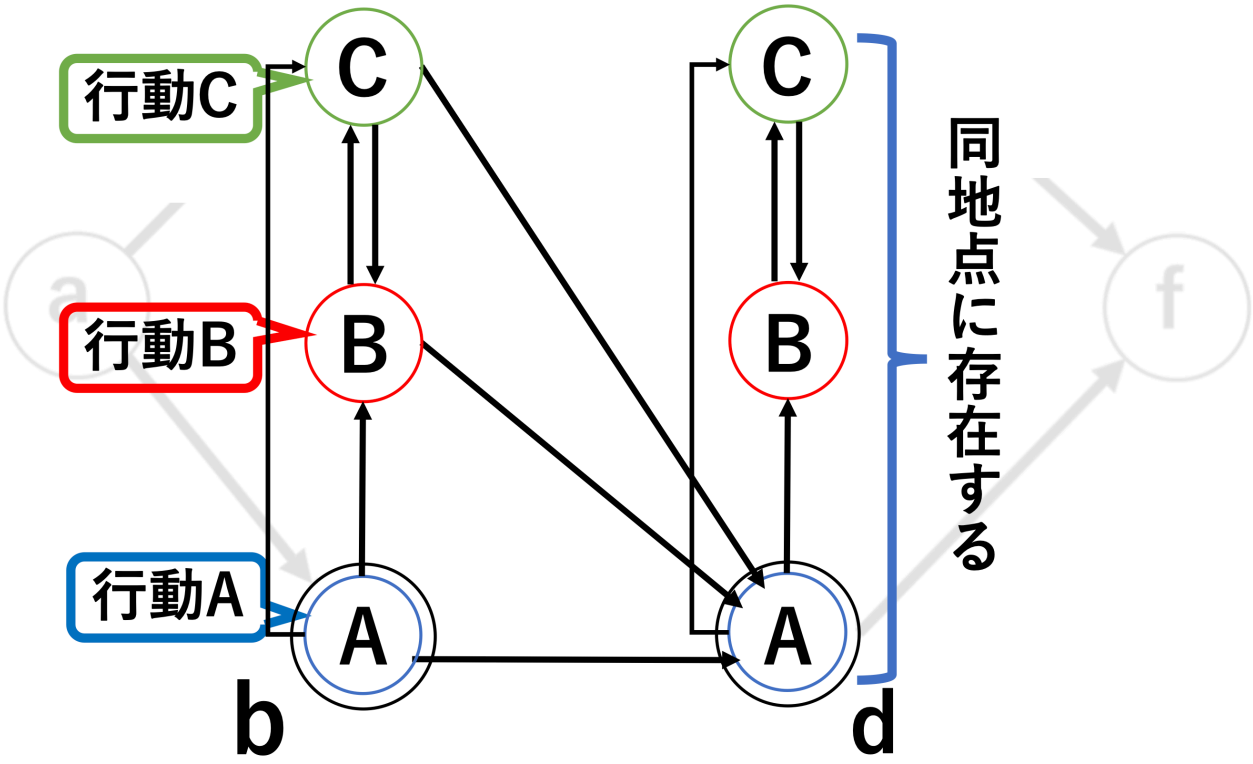

図 2.3　b 地点と d 地点における行動ノードとエッジ.

図 2.3 における行動の選択肢は、行動 A〜C としているが、行動ノードのイメージ図を示すためのものである。本手法では、各地点での移動を含む行動をノードとして扱うため、基本的に図 2.3 における行動 A には地点の遷移を行う「移動」を設定する。行動 B や行動 C における行動の選択肢は、例として、攻撃, 回復, アイテムの使用, 魔法などが考えられる。

ここで、グラフに関する補足説明をする。各地点の各行動ノードは、同じ場所にあり、上空に

ノードが設定されているわけではない。3 つの行動ノードは集約され、3 つのノードが合わさり 1 つの地点となる。行動をグラフとして表現する関係上、位置をずらして記述した。そのため、NPC が上空に移動しなければ、その他の行動ができないということはない。また、図 2.3 における各エッジの長さは意味を持たない。図として表現する関係上、長さに違いが生じてしまったものである。そのため、図 2.3 においてエッジの向きのみが意味を持つものとする。

各地点間の遷移に関して、別地点のノードを選択する場合は、任意地点の移動ノードを選択することで、任意地点まで移動し、行動 B や行動 C を行うことができる。先述通り、エッジの向いている方向の行動のみが選択できるものとする。このようにして、行動をノードとして扱い、経路探索の要領で行動経路を算出していく。

2.4 各行動ノードにおける重みについて

本手法の行動ノードにおけるエッジにはそれぞれ 3 つの重みを表すコストを設定する。各エッジにおける重みを図 2.1 を基にグラフで説明する。エッジ (i, j) における 3 種類の重みを (x_{ij}, y_{ij}, z_{ij}) とする。図 2.4 は、a から f 地点における 3 つの重みが付与されている状態を示している。図 2.4 では、地点を結ぶエッジのみ表示しているが、実際には各行動ノードにおけるエッジに 3 つのコストがそれぞれ設定されているものとする。

3 つのコストは、行動の選択肢に合わせてを設定する必要がある。移動のコストには、時間や距離といったコストを設定することが望ましい。その他の行動に関しても、適宜コストが表す要件を変更することで、条件を考慮する行動経路を算出することができる。

2.5 複数の制約条件について

制約条件については、2.4 節で示した 3 つのコストをもとに設定する。

- x の総和が最小

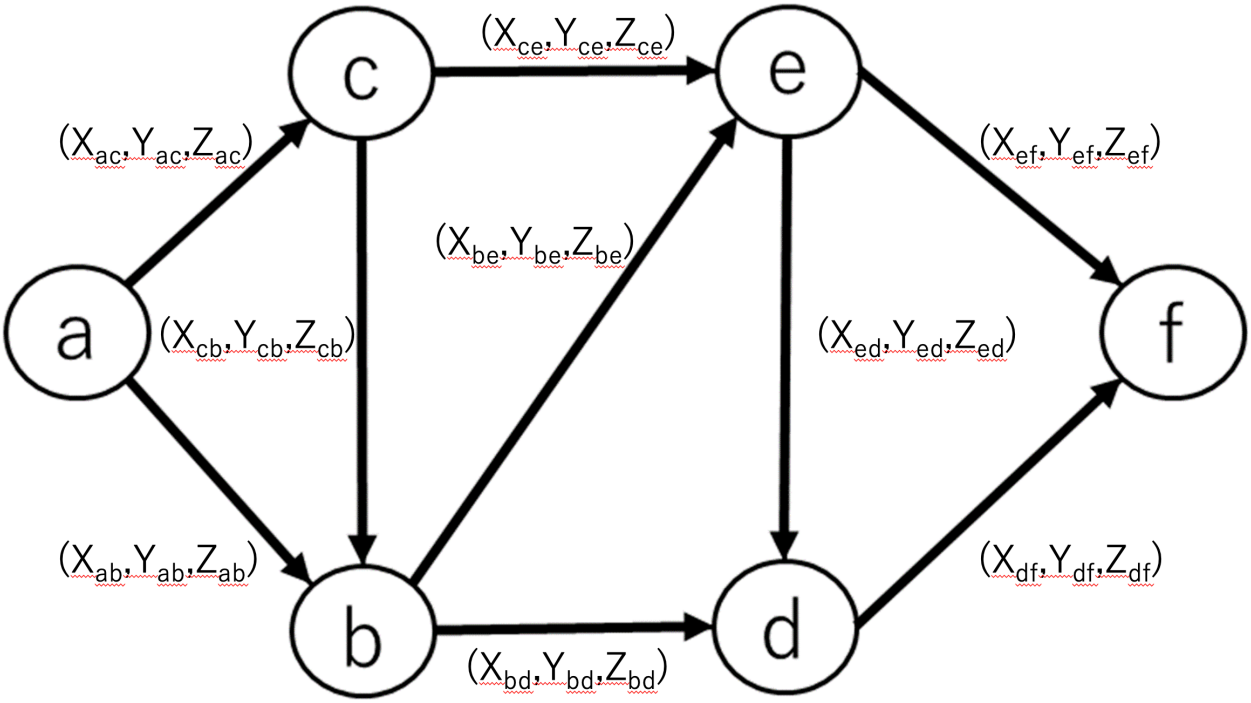

図 2.4　地点同士を結ぶエッジにおける 3 つの重みを示したグラフ.

- y の総和が最大
- z の総和が λ 以下

上記に示す 3 つ制約条件は、例であり最大・最小の条件や等式・不等式の条件は、必要に応じて変更することが可能である。また、λ の値も変更することが可能である。

上記の条件を数式を用いて説明する。この時、グラフ $G = (V, E)$ に対して、スタート地点を s, ゴール地点を $g(s, g \in V)$ とする。また、スタート地点 s とゴール地点 g を結ぶエッジの集合を $\bar{X} \subset 2^E$(E の冪集合) とする。これより、式 (2.1) が、「y の総和が最大」と「z の総和が λ 以下」を表し、式 (2.2) が、「x の総和が最小」を表す。制約付き最短経路問題では、式 (2.1) の前半部分、式 (2.2) のような最大・最小の算出を行うものを目的関数と呼び、式 (2.1) の後半部分のような等式・不等式の算出を行うものは制約関数と呼ぶ。制約付き最短経路問題では、本研究のように、目的関数が複数ある場合、多目的最短経路問題 [29] として扱う。この時、複数ある目的関数のうち、一つを目的関数として、それ以外を制約関数とすることで、解の算出を行う。そこで、

式 (2.1) の前半部分を目的関数とし、式 (2.1) の後半部分と式 (2.2) を制約関数として扱う場合を示したものが、式 (2.1) と式 (2.2) である。どちらの目的関数を優先するかは使用状況に応じて変更することが可能である。

$$\max_{X \in \bar{X}} \sum_{(i,j) \in X} y_{ij} \quad \text{s.t.} \sum_{(i,j) \in X} z_{ij} \leq \lambda \tag{2.1}$$

$$\min_{X \in \bar{X}} \sum_{(i,j) \in X} x_{ij} \tag{2.2}$$

探索の際は、各地点において式 (2.1) によって出た行動経路を記録していく。式 (2.1) における最大値は探索をしていく中で増えていく。そのため、記録してあった行動経路であっても、式 (2.1) の最大値より低い値となった行動経路は適宜削除する。式 (2.2) については以下のようにして扱う。任意の地点において、式 (2.1) により、y の総和が最大であり、かつ z の総和が λ 以下という条件が満たされる行動経路がいくつか記録される。これは、不等式によって行動経路の候補が 1 つに確定しないためである。そのため、式 (2.1) を満たす経路は、各地点において複数の行動経路の候補が算出される可能性がある。この時、式 (2.2) を適用してしまうと、各地点での経路候補が確定してしまい、その先で起こりうる状況に対応できないケースが発生していまう。例えば、事前に回復を行っていなかったために敵への攻撃ができないといったケースなどである。そのため、任意地点において記録されている経路候補と新たに算出された経路候補のコストの比較を行う。y の総和の値が同じであり、また z の総和の値も同じものであれば、その 2 つのうち式 (2.2) を満たす行動経路のみを記録し、もう片方の行動経路の記録を削除する。例として、任意の地点において、経路の各コストの総和が (10,3,2) という経路が記録されていたとする。同地点に対して、新たに経路の各コストの総和が (9,3,2) となる経路が導き出されたものとする。この時、新たに導き出された (9,3,2) となる経路を記録し、(10,3,2) という経路情報は削除する。このようにして、式 (2.2) を使用していく。探索終了時は、ゴール地点で式 (2.2) を満たす行動経路を遡ることで、複数条件を考慮した行動経路を算出することができる。

2.6　通過制限なし探索

2.6.1　通過制限ありと通過制限なしとの違いについて

2.2 節〜2.5 節までで示した方法は、通過制限ありの行動探索手法を示している。通過制限ありの行動探索とは、有向グラフで示したような地点間を移動できる方向が、決められているグラフでの探索のことを指している。以降、通過制限ありの行動探索を「通過制限あり探索」と呼称する。この時の通過制限ありとは、探索地点から任意地点 A,B それぞれに対してエッジがあり移動することが可能であるということを指しており、探索地点から任意地点 A にのみしか移動できないという意味ではない。探索地点から任意地点 A,B それぞれに移動することは可能であるが、任意地点 A から探索地点に移動した後に再度探索地点に戻り、任意地点 B に訪れることができないことを指している。そのため、通過制限あり探索での問題点が 2 つ発生する。

1. 使用するグラフが通過制限あり探索に対応したグラフにする必要がある
2. 迂回することができず、訪れられる地点に制限がある

第一の問題については、探索手法による影響によってグラフを対応させる必要がある。グラフを制限しなければ、目的関数を満たすために、同地点を何度も訪れるというような反復的な探索が発生する。その状況では、常に目的関数の値が更新され続けてしまうため、同地点を何度も探索してしまう。

第二の問題については、選択した行動経路によって、到達することができない地点が発生してしまうことである。これは、1 つ目の問題が大きく関わっており、制限されたグラフと探索手法によって起きる問題である。グラフによる制限によって、特定の地点に移動することができず、グラフと探索手法の制限によって、移動してきた道を戻るということができなかった。そのため、マップ上の全ての敵を倒す場合やアイテムを回収する場合などの迂回が必要となる状況では、通過制限あり探索では対応できないケースが発生する。通過制限あり探索では、グラフの制限内で

制約条件を満たす行動経路のみしか算出することができない。

そこで、通過制限なしの行動探索手法の提案を行う。以降、通過制限なしの行動探索を「通過制限なし探索」と呼称する。通過制限なし探索を行うために、探索地点から任意地点に対してのエッジと任意地点から探索地点に対してのエッジの両方を設定した、探索に制限のないグラフに変更する。グラフの各エッジにおけるコストは 2.4 節と同様のものとする。図 2.5 に通過制限あり探索の様子を、図 2.6 に通過制限なし探索の様子を示す。図 2.5 と図 2.6 に示す NPC は本手法を適用する NPC のことであり、敵は攻撃対象とする。NPC は敵を倒した数を最大化するということを目的関数とした時、通過制限あり探索では、グラフの制限により 1 体しか倒すことができないが、通過制限なし探索では、グラフの制限がないことで、2 体の敵を倒すことができるということを示している。

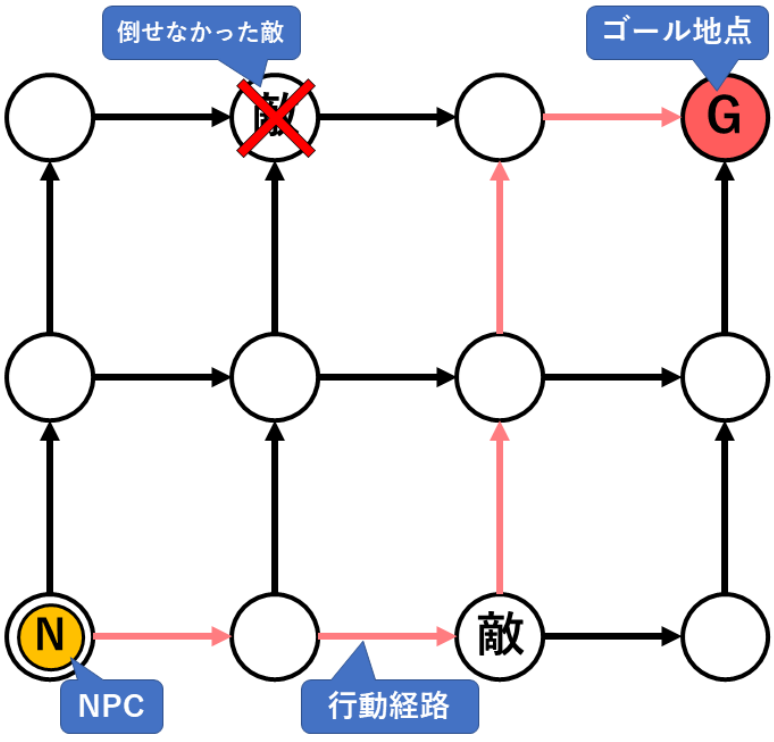

図 2.5　通過制限あり探索のイメージ図.

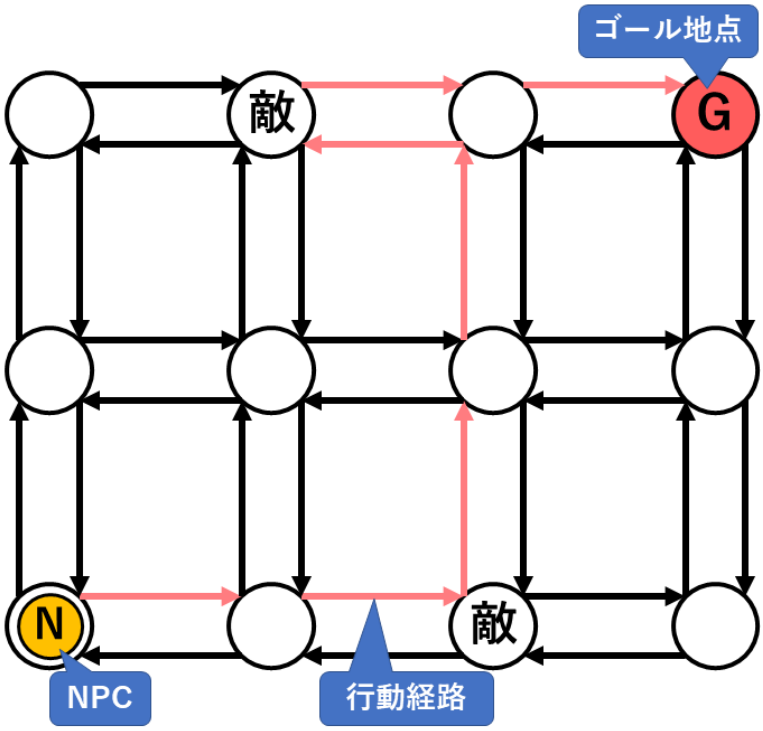

図 2.6　通過制限なし探索のイメージ図.

2.6.2　通過制限なし探索の手法について

通過制限なし探索は、通過制限あり探索に対して、以下のような変更点がある。

- 特定の経由する地点の訪問順を事前に決定しておくこと
- 目的関数の値が更新された地点において、更新前の経路候補を削除せずに残すこと

第一の変更点は、同地点に対する反復的な探索が起きることを防止する目的で設定した。グラフに制限がなくなるため、同地点を 2 度以上通過するケースが発生する。探索時に複数の行動経路の中で、特定の地点・行動を選択してきたかを判別することは困難である。そこで、経由する必要がある地点に対して、訪問順序を事前に決定しておくことで、目的関数の値が常に更新され続けるという状況にならないようにした。マップ上にいる敵への攻撃の順番や、アイテムの回収順など、特定の地点の訪問順を決めておくことで、反復的な探索を防止する。1 つ目の経由地点に

対して探索が終了すると次の経由地点に対して探索を行う。この時、1 つ目の経由地点のエッジは通常の地点のエッジと同様のコスト値に変更し、次の経由地点のエッジは、任意のコスト値に変更することで、次の経由地点に向かって探索ができる。コスト値を切り替えることで、同地点に対して反復的な探索が起きないようにしている。全ての経由地点を訪問した後は、ゴール地点に向かって探索を行うことで、行動経路を算出する。既存のゲームでも、時間の効率やキャラクターのレベルに合わせて任意の順番で特定の地点を訪れるということがある。この方法は、ゴールベース AI に分類される GOAP のような、ゴール地点からスタート地点までの行動を連鎖的に決定していくというイメージである。

第二の変更点は、ゴール地点からスタート地点まで行動経路を遡る際に、遡る経路がなくならないようにするために追加した。通過制限あり探索では、同じ地点の行動ノードは一度しか選択できず、また同地点を 2 度以上通ることがなかったため、目的関数の最大もしくは最小となる行動経路のみを記録しておけばよかった。しかし、通過制限なし探索では、2 度以上同地点を通る場合などがあるため、同地点に対して複数回探索が行われ、目的関数の値が更新された時、それよりも前に算出された記録も削除せずに残しておくこととする。これにより、同じ地点を複数回通る場合でも対応することができるようになった。

第 3 章

検証

3.1 実験環境

本研究では、Fine Kernel Tool Kit[30] を用いて、プログラムを作成した。表 3.1 に、検証を行う PC のスペックを示す。

表 3.1 検証用 PC スペック

OS	Windows 11
CPU	AMD Ryzen 7 2700 Eight-Core Processor 3.20 GHz
メモリ	16 GB
GPU	NVIDIA GeForce GTX 1650

3.2 エージェントの設定と敵について

本検証では、本手法を適用する NPC の「エージェント」とエージェントと敵対する NPC の「敵」を用意した。エージェントは、体力値が設定されており、体力値が 0 になるとゲームから離脱するものとした。検証でエージェントの行動の選択肢は、「移動」,「攻撃」,「回復」とした。

移動を選択すると、エージェントが存在する地点から対象の地点に移動することができるものとした。

攻撃を選択すると、敵の有無に合わせて以下のようになるようにした。敵がいる地点で攻撃を行った場合は、敵を倒したことになり、報酬と敵からのダメージを受けるものとした。敵がいない地点で攻撃を行った場合は、報酬とダメージはないものとした。

回復を選択すると、受けたダメージの総量を減らすことができるものとした。エージェントが生存し続けるためには、体力値の管理が重要となる。ダメージ総量が多くなればなるほど、体力値が削れていることになるため、ある一定値以下のダメージ総量であることが、体力値を 0 にさせないための条件となる。本検証では、ダメージの総量を回復を選択することで減少させるため、受けたダメージ量以上に回復はできないとした。つまり、0 より低いダメージ総量になることはないものとした。

3.3　検証に用いるコストと制約条件について

本手法のエッジにおける、3 つのコストを (時間, 攻撃ポイント, ダメージ) とした。コストはそれぞれ、

- 時間…行動を実行するのにかかる時間
- 攻撃ポイント…敵を倒した報酬
- ダメージ…敵から受けるダメージ量

を表す。

各行動におけるエッジの重みを図 2.3 に示す b 地点と d 地点に表示した様子を図 3.1 に示す。任意の地点に敵がいる場合は「敵あり」の重みを使用し、敵がいない地点においては「敵なし」の重みを設定して探索を行った。敵ありの重みは、敵がいる地点のため、攻撃ポイントに敵を倒した報酬を、ダメージには敵から受けるダメージ量を設定した。敵なしの重みは、敵がいない地点のため、通常の地点と同様に攻撃ポイントとダメージは 0 とした。回復をするとダメージ総量が減るため、回復のエッジのダメージコストには -1 を設定した。

制約条件については、以下の 3 つを設定した。

- 時間が最短
- 攻撃ポイントの総和が最大
- ダメージの総和が λ 以下

λ はエージェントの受けられるダメージの最大許容値を示しており、体力値より小さな値が入る。ダメージの総和が λ 以下であることが体力値を 0 にしないことの条件とした。

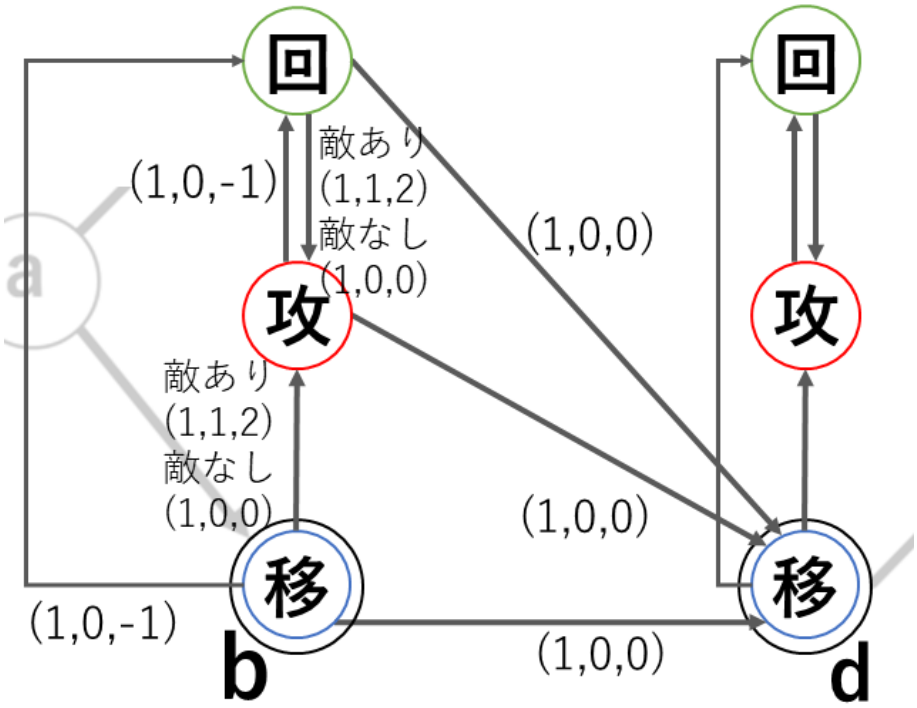

図 3.1　b 地点と d 地点における各行動ノードのエッジの重み.

3.4　検証概要

検証では、本手法を用いることで、エージェントの移動経路の探索と行動決定を同時に行い、かつ複数の制約条件を満たす行動経路を算出することができるかを様々な探索状況で検証した。それぞれの探索状況において、長期的目標でもあるエージェントがゴール地点に到達することと、服数の制約条件を満たす行動が算出できることを確認することが検証の目的となる。探索を行うグラフに関しては、図 2.1 と同様のグラフを用いた 6 地点のグラフと縦横に 10 × 10 地点を配置した計 100 地点のグラフの 2 種類での実験を行った。3.5 節に、本手法の概要を理解してもらうために 6 地点のグラフにおいて通過制限あり探索による検証を行った。3.6 節では、敵を倒すことを優先事項とした際の通過制限あり探索と通過制限なし探索の検証を行った。状況が変わることでの行動算出結果の差異を示すために、体力値上限を 5 とするもの、体力値上限を 7 とするもの、初期体力値が減少した状態での 3 パターンでの検証を行った。また、3.7 節では、最短経路を優先

事項とした際の結果について記述する。

本手法では、各グラフにスタート地点とゴール地点を設定し、ゴール地点にたどり着くことを長期的な目標とした。スタート地点とゴール地点はそれぞれ、任意の地点の移動ノードとした。

実行画面についての説明をする。黄色の球体はエージェントを表しており、行動探索の結果に合わせて行動する。画面左上にはエージェントの行動をテキストで示しており、3 つの行動が適宜切り替わるようにした。

3.5 手法の概要説明のための検証

本節の検証は、本手法の概要を理解してもらうために各行動をノードとして表示し、エージェントがどのように行動を選択していくのかを示す。また、通過制限あり探索による算出結果を示す。

3.3 節にて紹介したのλを以下の通りに設定した。本節では、エージェントの体力値の最大は 3 とし、λを 2 として、探索を行った。また、エージェントの体力値は 3 の状態でプログラムを起動した。

図 2.1 における a 地点をスタート地点とし、f 地点をゴール地点として行動経路を算出した。この時、a 地点の移動ノードをスタートのノードとし、f 地点の移動ノードをゴールのノードとした。c 地点と d 地点のそれぞれに敵がいる状態を図 3.2 に示す。グラフにおける表示方法として、スタート地点を淡青色、ゴール地点を淡赤色で表示し、敵のいる地点を黒色で表現した。

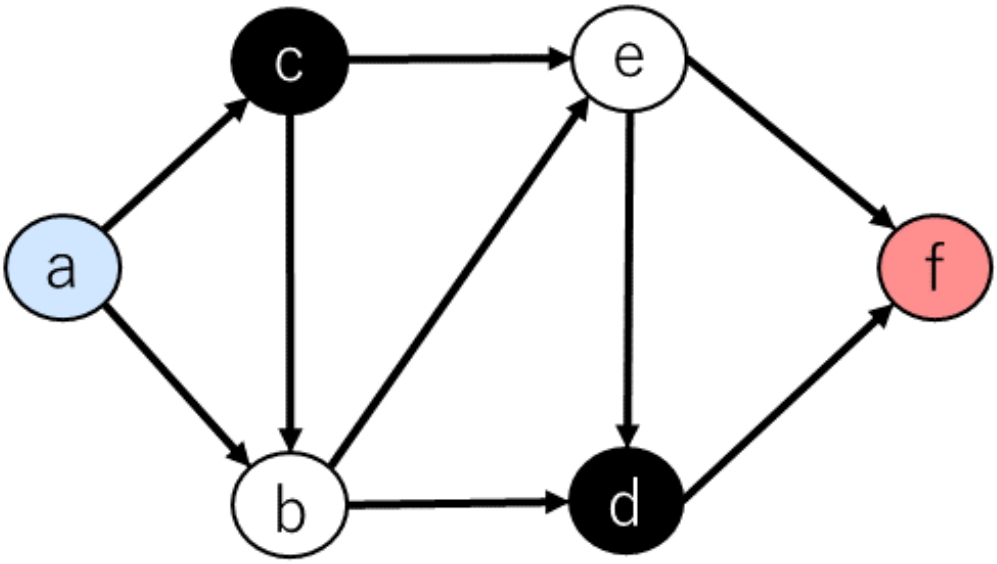

図 3.2　c,d 地点に敵がいるイメージ図.

本手法の概要をよりわかりやすくするため、本節における検証では各地点における行動ノードを独立させて表現を行った。図 3.3 に、6 地点における実行画面を示す。しかし、スタート地点やゴール地点、敵の位置などは図 3.2 の通りだが、実行結果にはそれらの位置において色による表示を行わなかった。

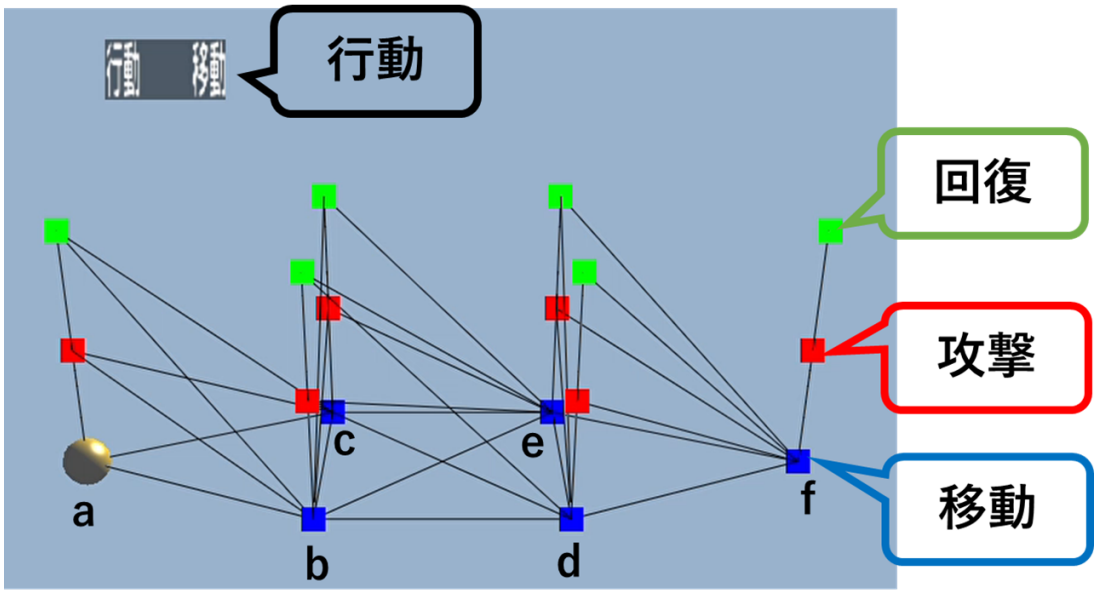

図 3.3　プログラムの実行画面.

図 3.3 の各地点における移動ノードの下に a 地点から f 地点を記す。また、青色のノードが移動ノード、赤色のノードが攻撃ノードを、緑色のノードが回復ノードを示している。縦に並ぶ移動、攻撃、回復の 3 つのノードを同地点に存在するものとして探索を行っていく。a 地点の移動ノードにある黄色い球体がエージェントを示しており、行動経路を遷移する。図 3.3 における、c,d 地点のそれぞれの攻撃ノードへのエッジは、図 3.1 の敵ありの時の重みを設定した。それ以外の地点の攻撃ノードへのエッジに関しては敵なしの重みを設定した。図 3.3 を実行した際、条件を考慮した経路として選択した経路を以下に示す。また、どの地点の行動ノードを選択したのかを示す記述方法として、a 地点における移動ノードを選択したのちに a 地点の攻撃を選択する場合、a(移動) → a(攻撃) と記述する。

まず、エージェントはスタート地点である a 地点から敵がいる c 地点へ移動した。c 地点までは、a(移動) → c(移動) → c(攻撃) → c(回復) という行動経路を辿った。その様子を図 3.4 に示す。

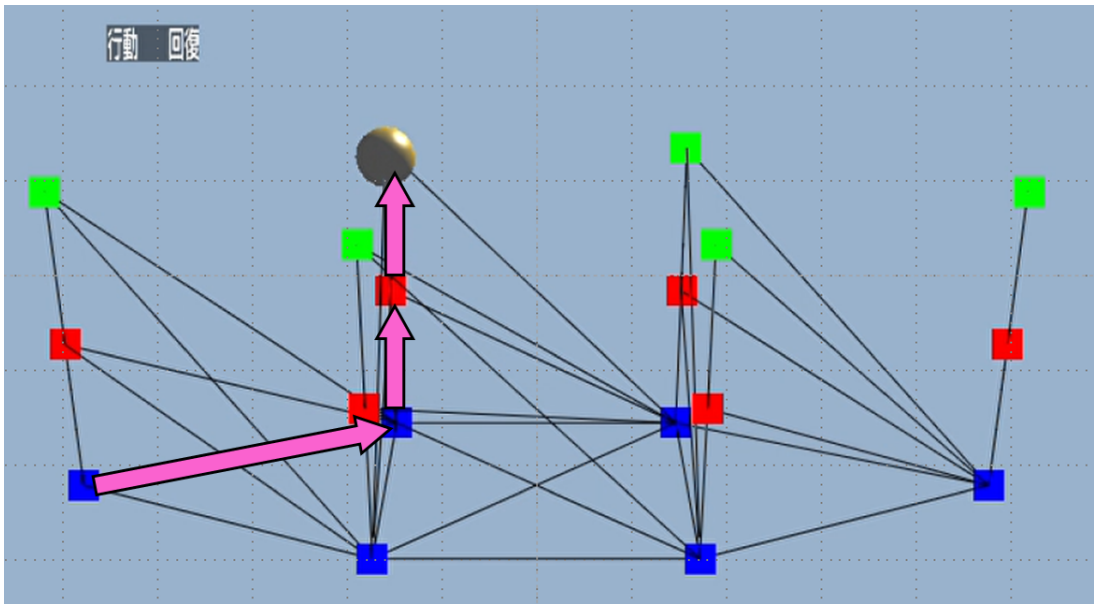

図 3.4　a 地点から c 地点への遷移した状態.

次に、c 地点から e 地点へと移動した。e 地点までは、c(回復) → e(移動) という行動経路を辿った。e 地点からは、そのまま d 地点の移動ノードへ遷移した。d 地点までは、e(移動) → d(移動) → d(回復) → d(攻撃) という行動経路を辿った。その様子を図 3.5 に示す。

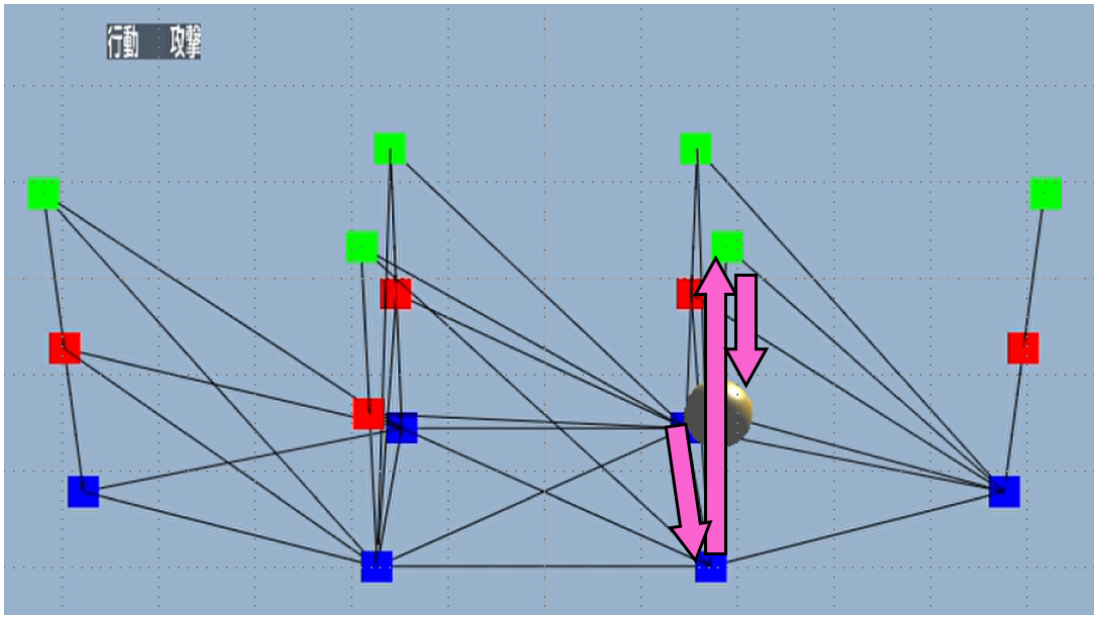

図 3.5　e 地点から d 地点への遷移した状態.

最後に、d 地点から f 地点までは、f 地点の移動ノードに遷移して行動が終了した。

結果として、a(移動) → c(移動) → c(攻撃) → c(回復) → e(移動) → d(移動) → d(回復) → d(攻撃) → f(移動) という行動経路が算出された。上記の経路を図 3.6 に示す。

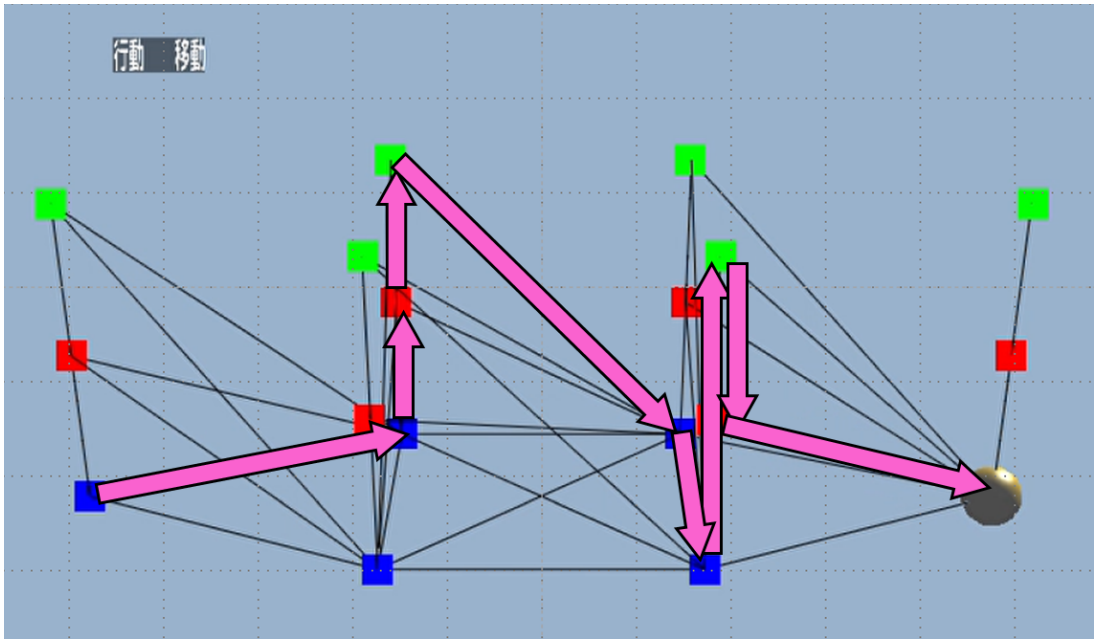

図 3.6　a 地点から f 地点までの算出された行動経路.

3.6　敵を倒すことを優先事項とする検証結果

3.6.1　体力値上限を 5 とする通過制限あり探索での検証

本項では、敵を倒すことを優先事項とした際の通過制限あり探索による算出結果を示す。3.3 節にて紹介したのλを以下の通りに設定した。本項では、エージェントの体力値の最大は 5 とし、λを 4 として、探索を行った。また、エージェントの体力値は 5 の状態でプログラムを起動した。

実行画面を図 3.7 に示す。実行画面についての説明を行う。100 地点は白色の四角で表され、地点同士は黒色のエッジで繋がっている。スタート地点を淡青色、ゴール地点を淡赤色で表示し、それぞれマップの左下と右上に設定した。

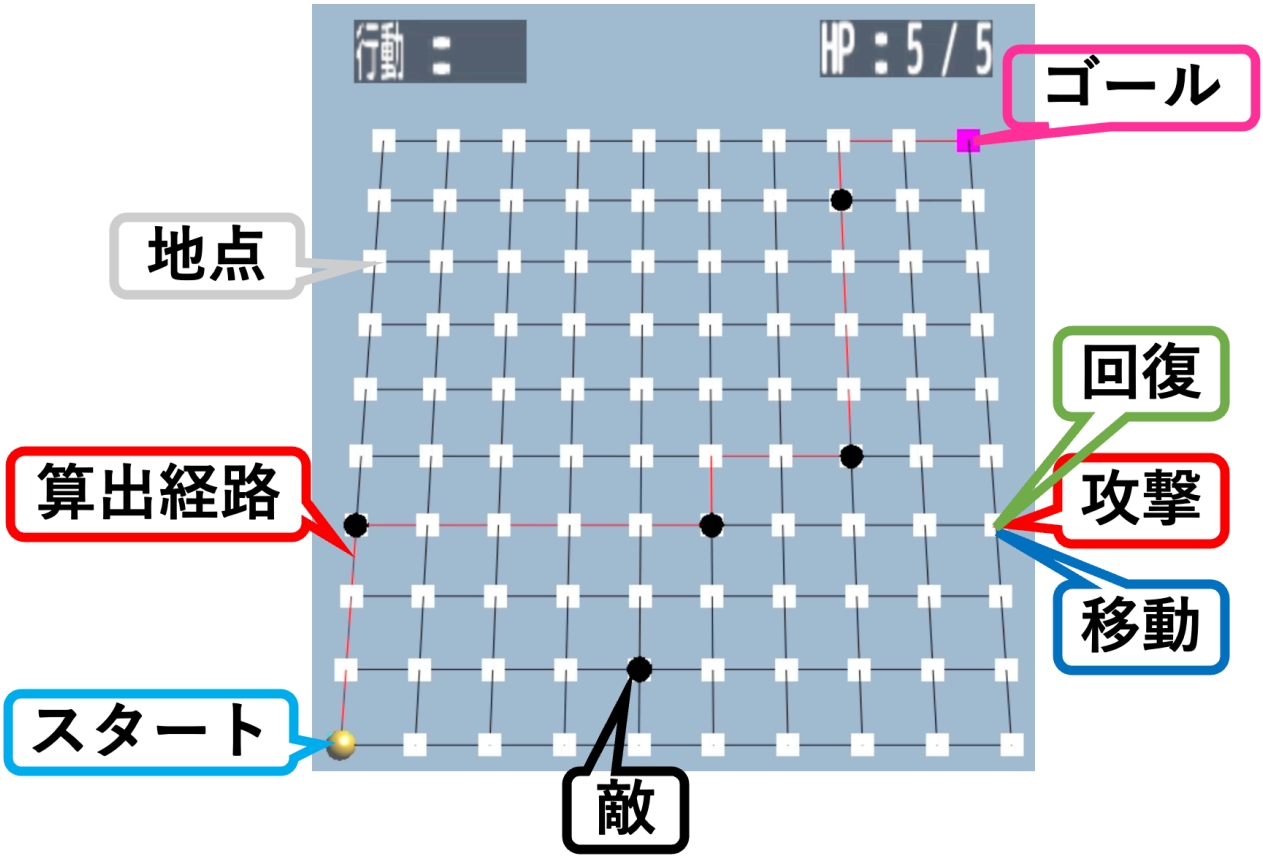

図 3.7　敵を倒すことを優先事項とする通過制限あり探索における実行画面.

本項では、3.5 節と比べて、6 つの違いがある。第一に、各頂点にある行動ノードを同地点に集

約し、地点として白色のノードのみを表示するようにしたことである。第二に、エージェントが地点を遷移するたびに行動経路の再探索が行われるようにしたことである。3.5 節の検証では、敵の位置は変化せず、エージェントのみが行動していたが、実際のゲームでは敵も行動をするため、このような処理を追加した。第三に、敵の表示を追加したことである。先述のように、敵も行動するためエージェントが地点を移動するたびに敵も地点を移動するようになっている。第四に、エージェントの体力値を表示するようにした。画面右上にはエージェントの体力値を”現在の体力値/最大値の体力値”の形式で表示しており、敵からダメージを受けると現在の体力値が変化するようになっている。第五に、どの行動経路を辿ろうとしているのかを示すためにエッジを赤色で表示したことである。これにより、エージェントと敵が遷移するたびに赤色の算出経路が更新されるため、どの経路を辿るのかを視覚的表現した。第六に、エージェント自体の色を行動に合わせて変化するようにしたことである。エージェントの各行動における色の変化は、図 3.8 に示す。これにより、エージェントが現在どの行動をしているのかを左上を見ることなく視覚的に理解できるようにした。

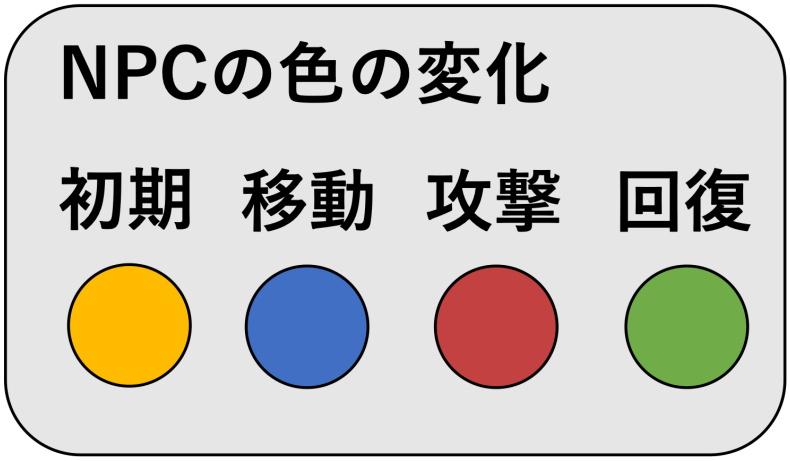

図 3.8　エージェントの各行動における色の変化.

次に敵の移動の順路について説明する。検証では、マップ上に敵を 5 体配置し、5 体中 3 体は地点を遷移し、2 体は静止しているものとした。敵の移動は、エージェントの行動が完了した後で、

一斉に行った。図 3.9 は、エージェントと敵の両者が 1 地点ずつ移動した様子を示す。また、図 3.9 は、移動する敵の順路を矢印にて示す。矢印のない敵に関してはその場で動かない敵とした。

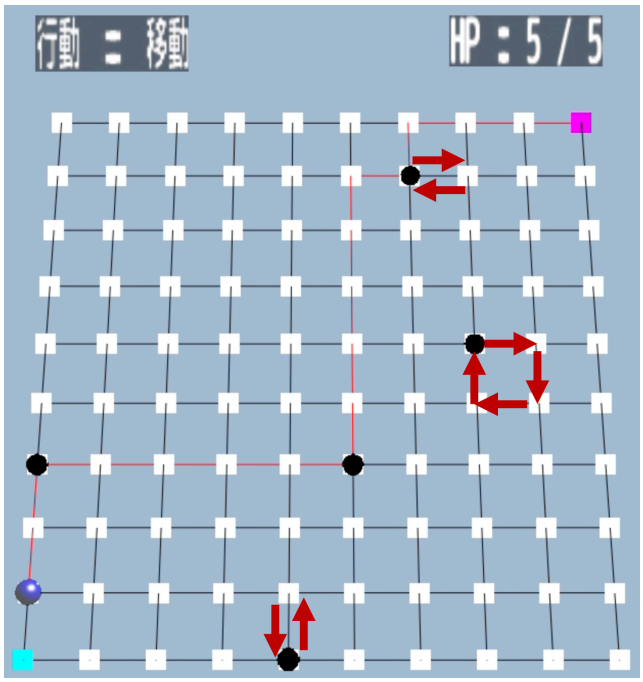

図 3.9　エージェントと敵が移動した様子.

続いて、探索経路の結果について説明する。まず、エージェントは、図 3.9 における一番左側にいる敵に向かって移動した。図 3.10 に、敵の地点まで移動した様子を示している。図 3.11 に、敵に攻撃した様子を、図 3.12 に、攻撃後に回復をした様子を示している。図 3.13 に、敵を撃破したことで、画面上からは敵が消滅し、次の地点に向けて移動を始めたエージェントの様子を示す。敵は消滅すると、その地点から黒色の球体は消滅し、通常の地点と同様の扱いとした。

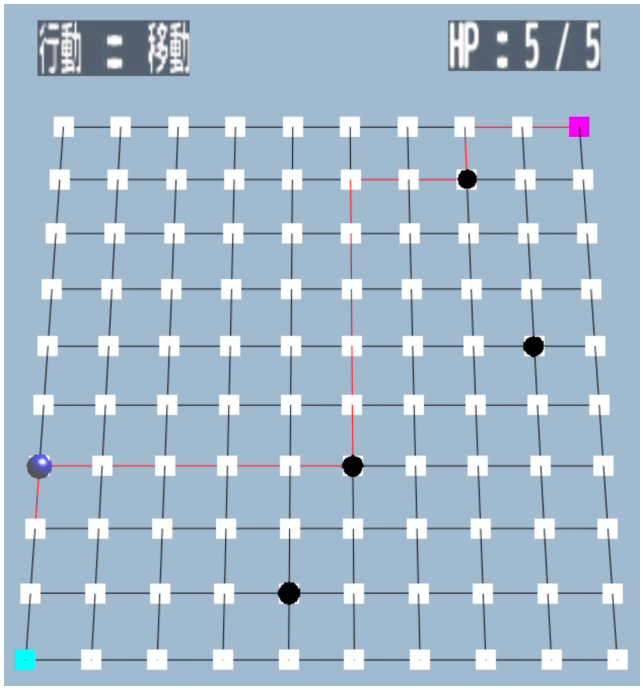

図 3.10　一番左側にいる敵の地点まで移動した様子.

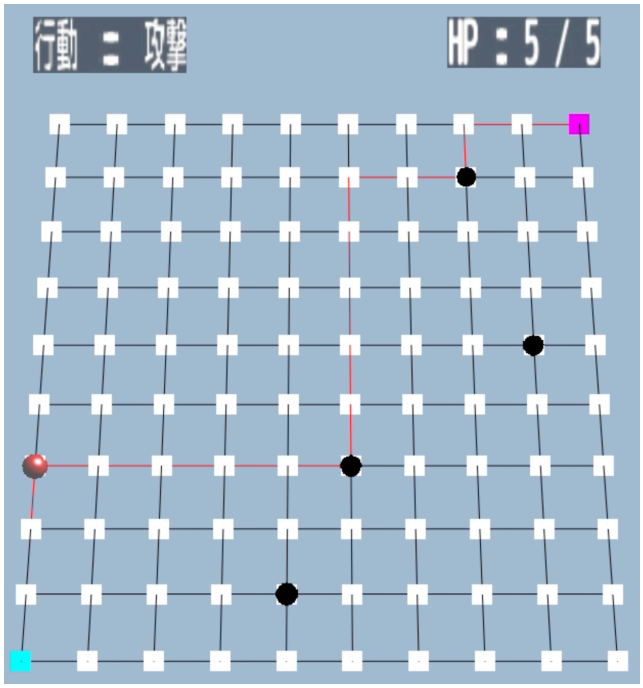

図 3.11　一番左側にいる敵を攻撃した様子.

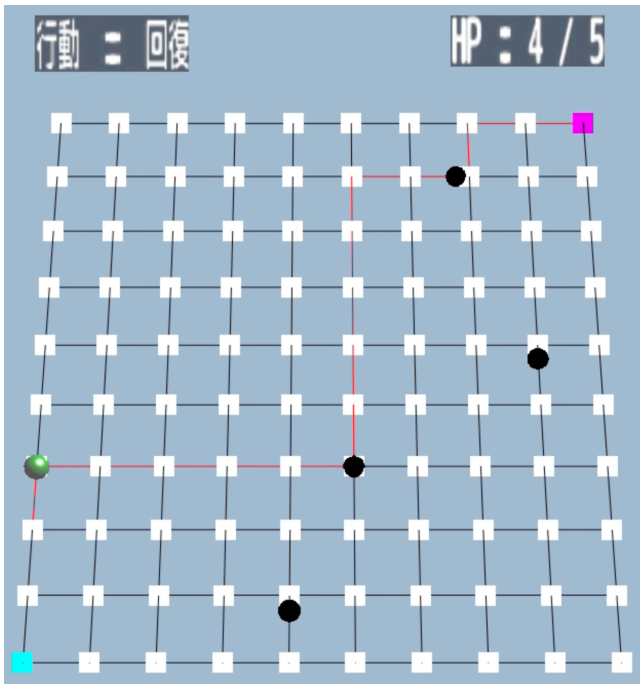

図 3.12　攻撃後に回復をした様子.

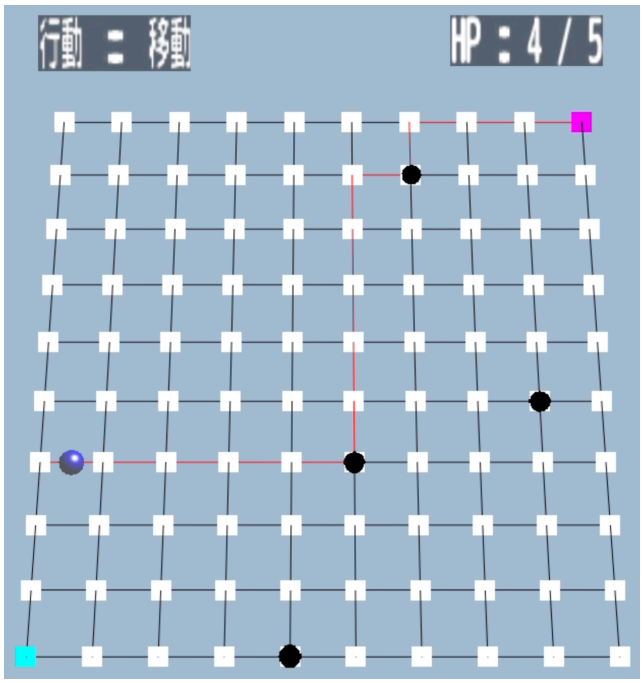

図 3.13　敵が消滅し、次の地点へ移動を開始したエージェントの様子.

次にエージェントは、画面中央にいる敵を目指し移動した。エージェントが敵の地点まで移動し、その後攻撃した様子を図 3.14 に示す。

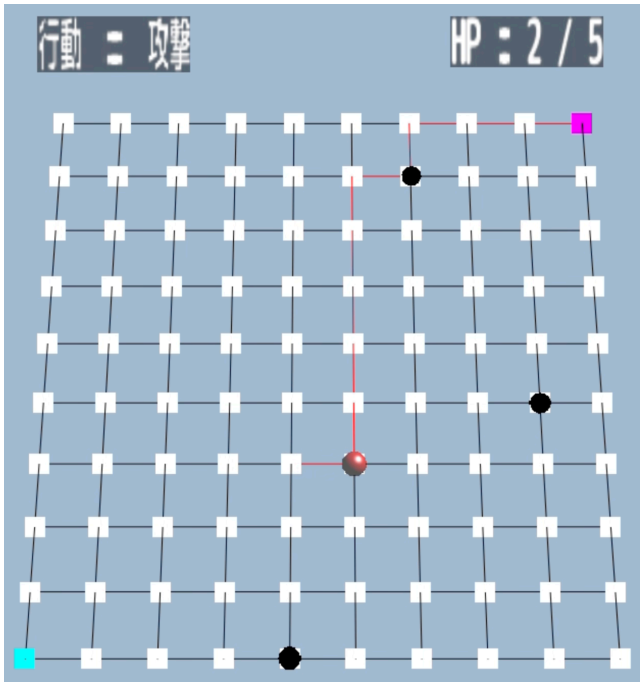

図 3.14　敵地点まで移動し、攻撃をした様子.

この段階で、体力値は 2 となっており、この状態で、敵に攻撃をしてしまうと、体力値が 0 になってしまうため、次の地点では回復を行った。次の地点に移動し、回復を行った様子を図 3.15 に示す。図 3.15 の段階で、エージェントの右側に位置する敵と一番上側にいる敵の両方を倒す経路を算出するが、エージェントが回復した後に敵が移動してしまうため、2 体の敵を倒すことができず、一番上に位置する敵を倒すために移動した。図 3.16 は、右側の敵を倒すことをやめ、一番上側にいる敵を倒しに向った様子を示す。

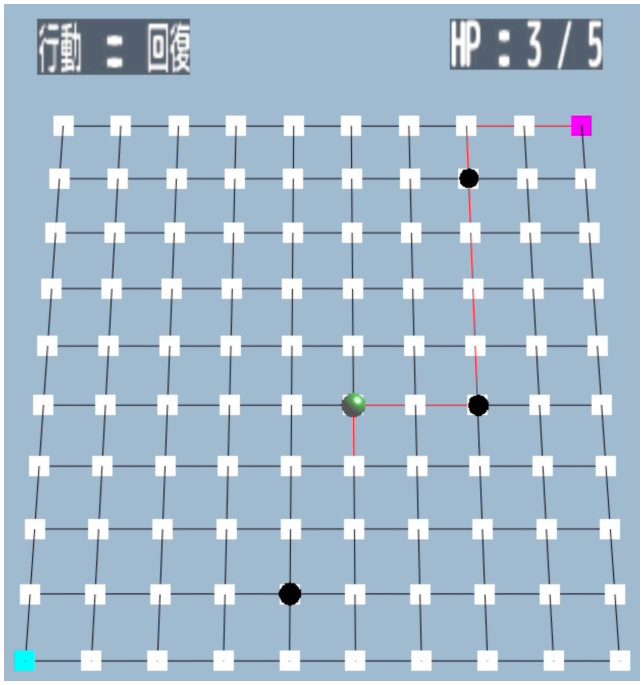

図 3.15　次の地点に移動し、回復をした様子.

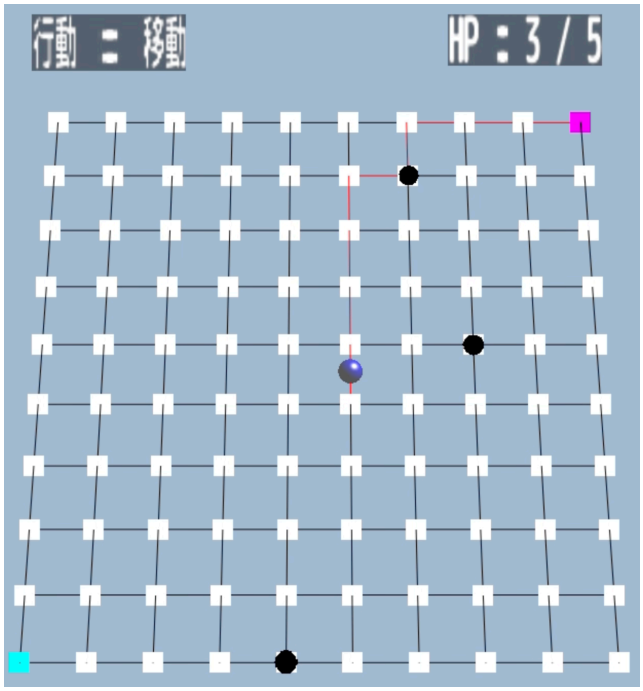

図 3.16　一番上側の敵に向けて移動する様子.

この後は、一番上側の敵の地点まで移動した。図 3.17 は、敵の地点まで移動し、攻撃をした様子を示す。敵に到達した時点で、体力値は 3 であるため、回復を行わずに攻撃をすることが可能

となった。攻撃をした後は、体力値が 1 の状態でもその後に攻撃することがないため、回復を行わず、ゴール地点に移動した。図 3.18 は、ゴール地点に到達した様子を示す。

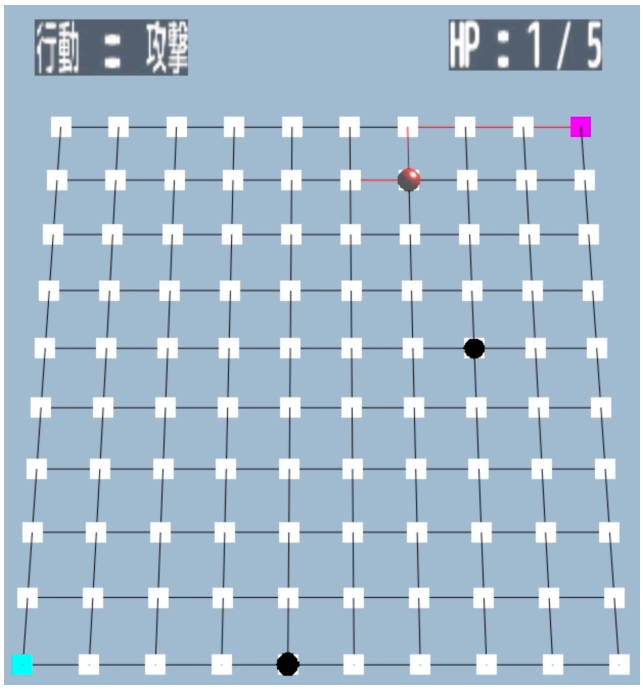

図 3.17　一番上側にいる敵の地点まで移動した様子.

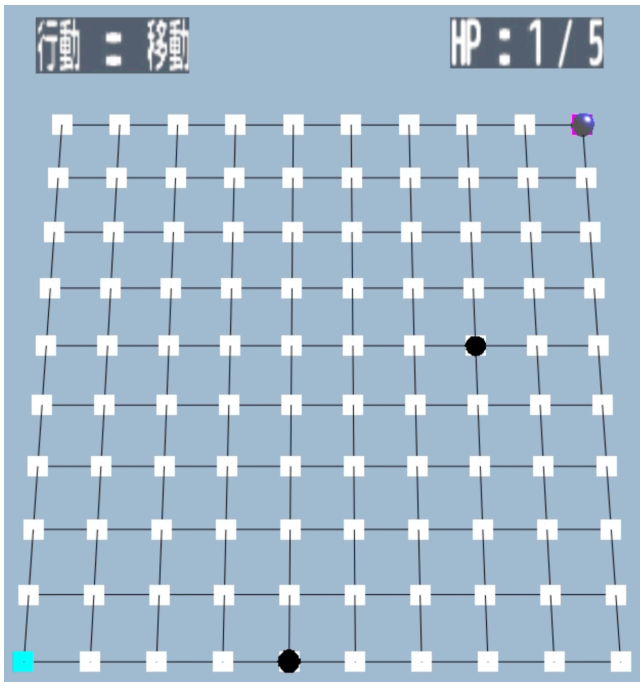

図 3.18　ゴール地点に到達した様子.

3.6.2　体力値上限を 5 とする通過制限なし探索での検証

本項の検証では、通過制限なし探索を 3.6.1 項と同様の条件で実験を行った。また、敵の状態に関しても同様の内容である。

次に、倒す敵の順番の決め方について説明する。今回の検証で敵の倒す順番は、式 (3.1) によって、求めた値によって決めた。式 (3.1) における $D_a(i)$ はエージェントから敵 i までの距離を表しており、$D_g(i)$ はゴール地点から敵 i までの距離を表している。$D(i)$ は敵 i の計算結果を表す。すべての敵に対して計算を行い、D 値の大きな敵から順に倒すものとした。

$$D(i) = D_g(i) - D_a(i) \tag{3.1}$$

実行画面を図 3.19 に示す。

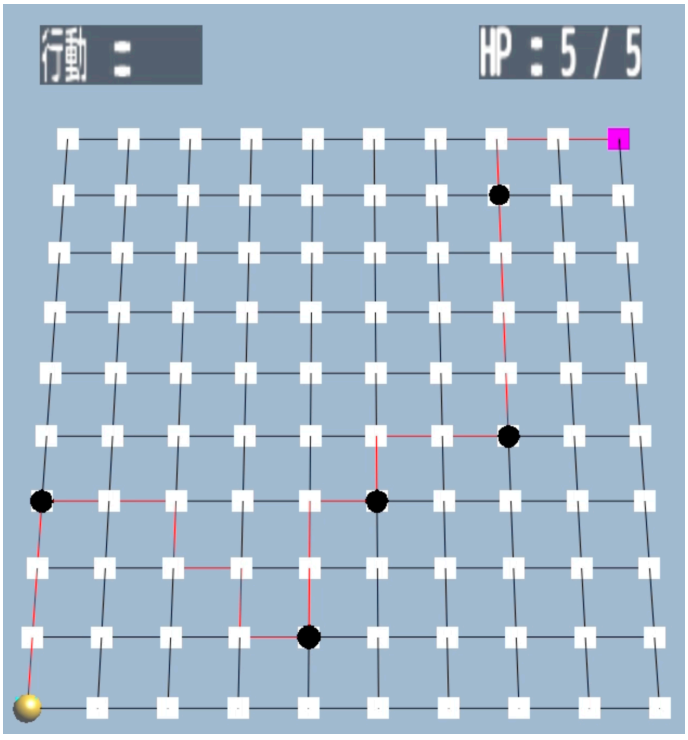

図 3.19　検証実験の実行画面.

エージェントは行動経路に沿って移動していき、図 3.19 におけるエージェントの上側に位置する敵を攻撃した様子を図 3.20 に示す。この時、攻撃したことで一時的に体力値が 3 となり、その

後、回復を行ったため体力値が 1 回復して 4 となった。図 3.20 では、エージェントの行動が完了し、敵も移動が完了した状態である。

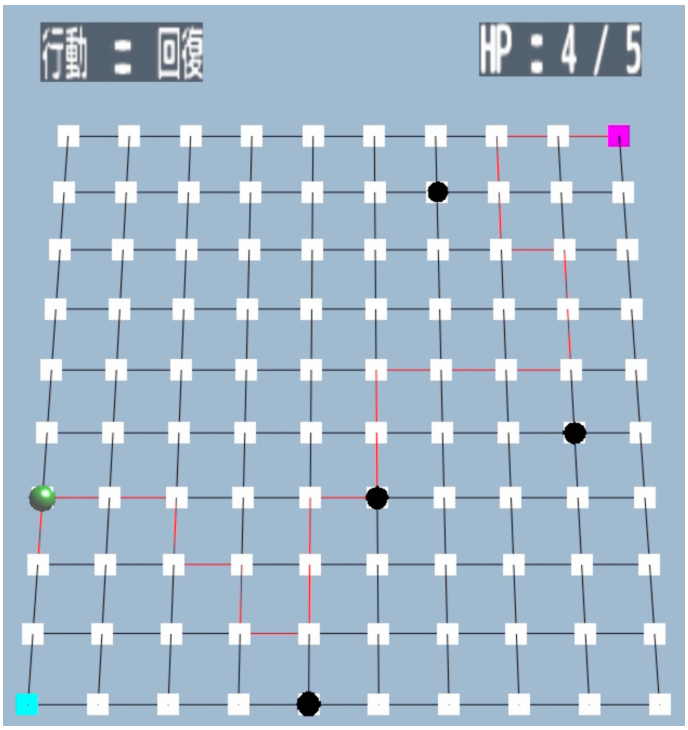

図 3.20　エージェントの行動が完了し、敵が移動した様子.

図 3.20 の後、エージェントは図 3.21 に示す新たな行動経路を算出し、移動をした後に回復を行い体力値を 5 にして、次の敵に向けて移動した。

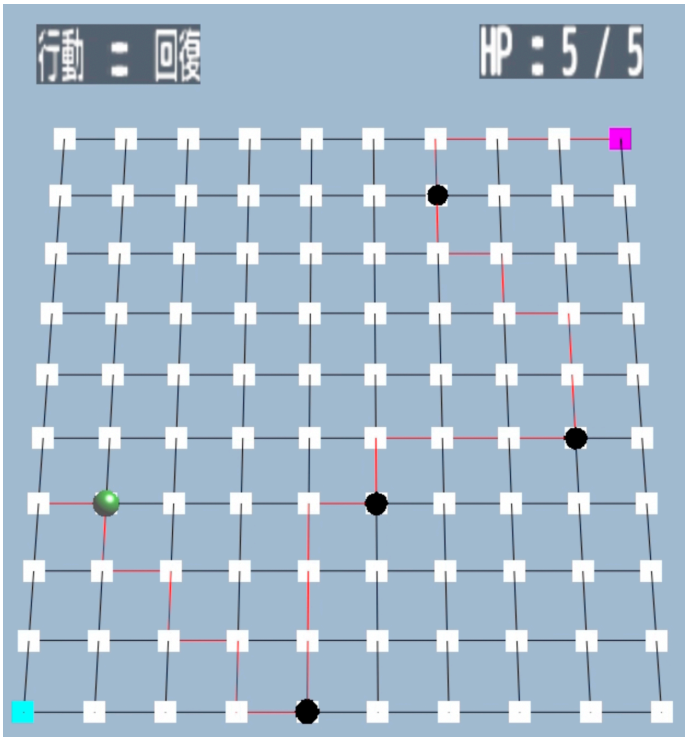

図 3.21　新たな行動経路を算出し、回復をした様子.

2,3 体目は 1 体目の敵を倒した時と同様に、攻撃した地点で回復を行い、その隣の地点でも回復を行うという流れで行動を行った。この時、体力値の変化としては、攻撃をして 3 となり、同地点で、回復をして 4 となった。別の地点に移動して、回復を行うことで、体力値は 5 になった。そのため、2,3 体目の敵を倒す時はどちらもダメージの総量が 0 の状態で攻撃を行った。図 3.22 は、2 体目の敵を倒して回復を行った様子を示す。同様に、図 3.23 は、3 体目の敵を倒して回復を行った様子を示す。

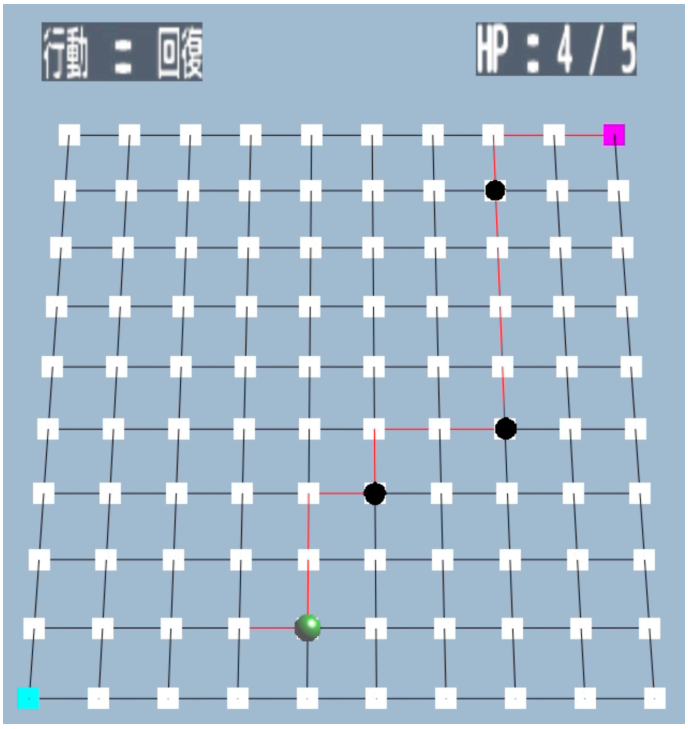

図 3.22　2 体目の敵を倒し、回復をした様子.

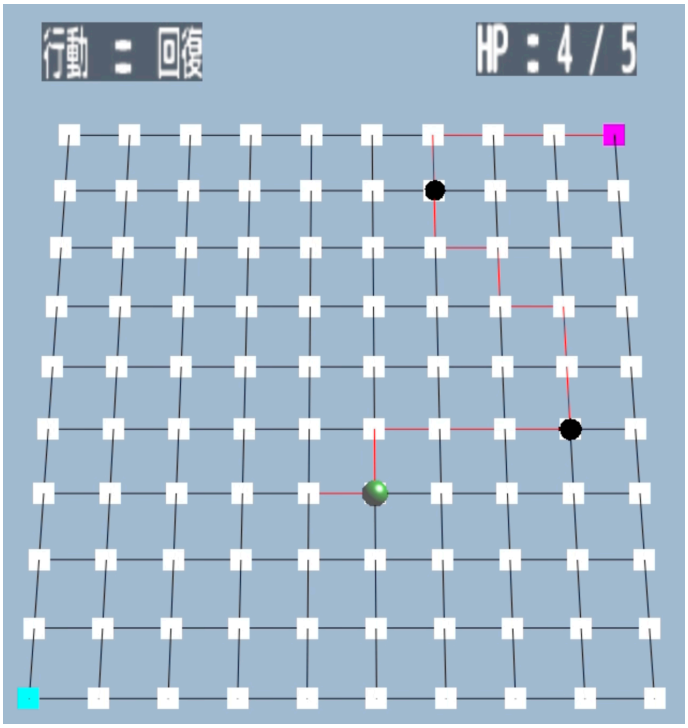

図 3.23　3 体目の敵を倒し、回復をした様子.

3 体目の敵を倒した後は、体力値を 5 の状態にして 4 体目の敵の地点に向けて移動した。

4 体目の敵に向かって移動している最中に同じ地点を 2 度通るケースが発生した。図 3.24 は、2 度同じ地点を通るケースを示す。通過制限あり探索では、このような 2 度同じ地点を通るケースに対応することができなかった。通過制限なし探索では、このケースにも対応して行動経路を算出することができた。

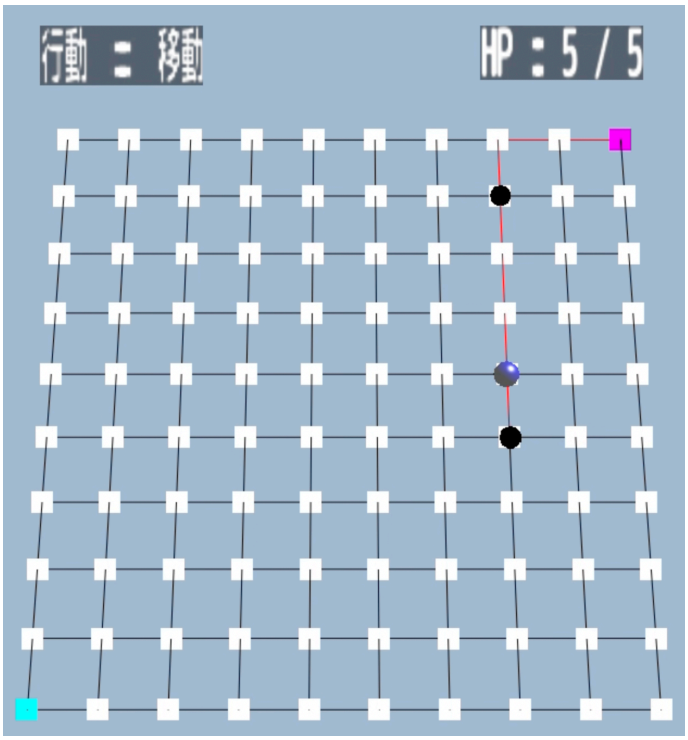

図 3.24　同じ地点を 2 度通るケース.

図 3.25 は、このケースに対応して算出した行動経路の結果をもとに 4 体目の敵を倒した様子を示す。

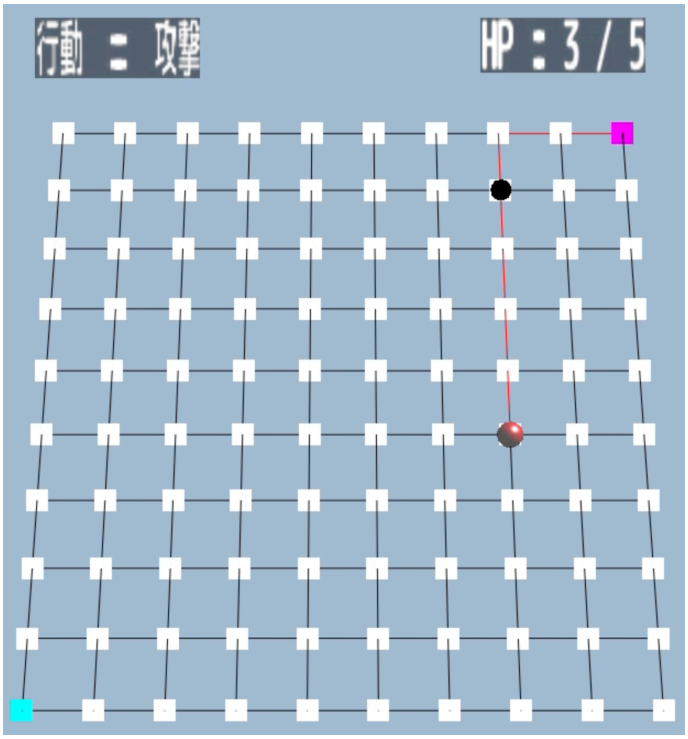

図 3.25　4 体目の敵を倒した様子.

4 体目の敵を倒した時、体力値は 3 になるものの、残りの敵は 1 体であり、回復を行わなくても攻撃ができるため、このまま移動を開始した。4 体目の敵を倒した後、エージェントは図 3.24 と同じ地点に戻り、最後の敵を倒しに移動した。図 3.26 は、5 体目の敵を倒した様子を示す。

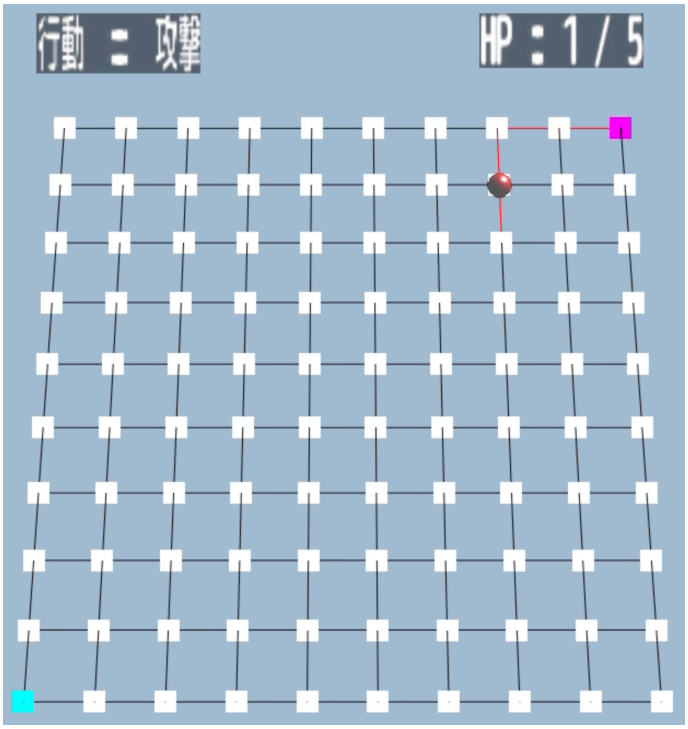

図 3.26　5 体目の敵を倒した様子.

5 体目の敵を倒した後の体力値は 1 となり、その後敵を倒すことはないため、回復を行わず、ゴール地点まで移動した。このように、2 度同じ地点を通るケースにも対応し、すべての敵を倒して最終目標であるゴール地点まで到達することができた。図 3.27 にゴール地点に到達した様子を示す。

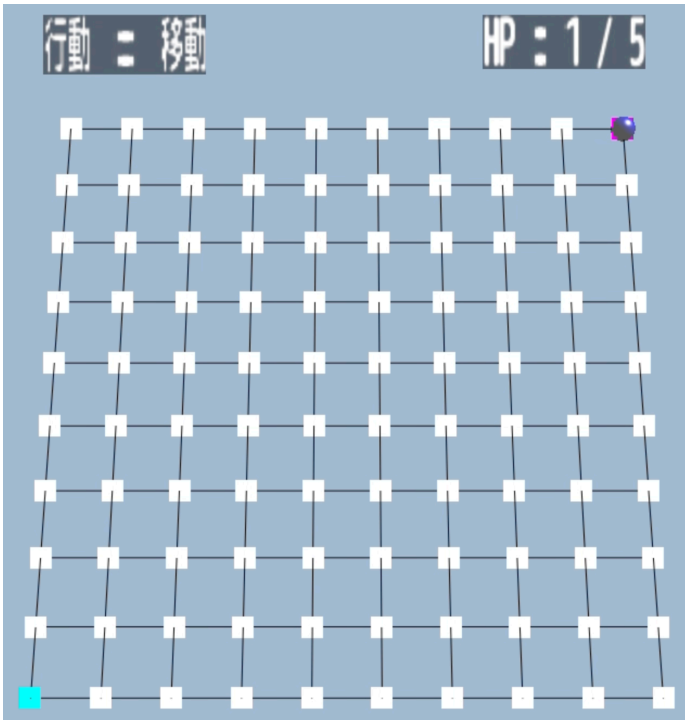

図 3.27　ゴール地点に到達した様子.

ゴール地点に到達した時の体力値は 1 となるため、体力値の条件も満たしていることがわかった。

3.6.3　体力値上限を 7 とする通過制限あり探索での検証

3.6.1 項の体力値上限を 5 とする通過制限あり探索の検証に対して、体力値上限を 7 として検証を行った。この時、λの値は 6 とした。図 3.28 に、体力値上限を 7 とする通過制限あり探索での実行画面を示す。

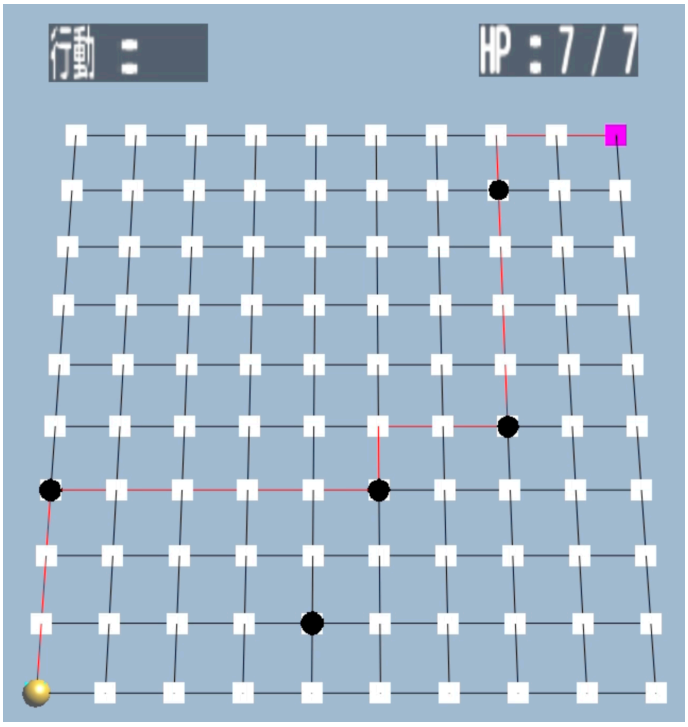

図 3.28　体力値上限を 7 とする通過制限あり探索での実行画面.

図 3.29 に、マップ左端の敵を倒した後に回復を行わずに、移動をするエージェントの様子を示す。3.6.1 項の体力値上限を 5 とする通過制限あり探索の検証では、マップ左端の敵に対して攻撃を行った後に同地点で回復を行っていたが、体力値が 7 に上昇したことで、回復を行わずに次の地点に移動するようになった。

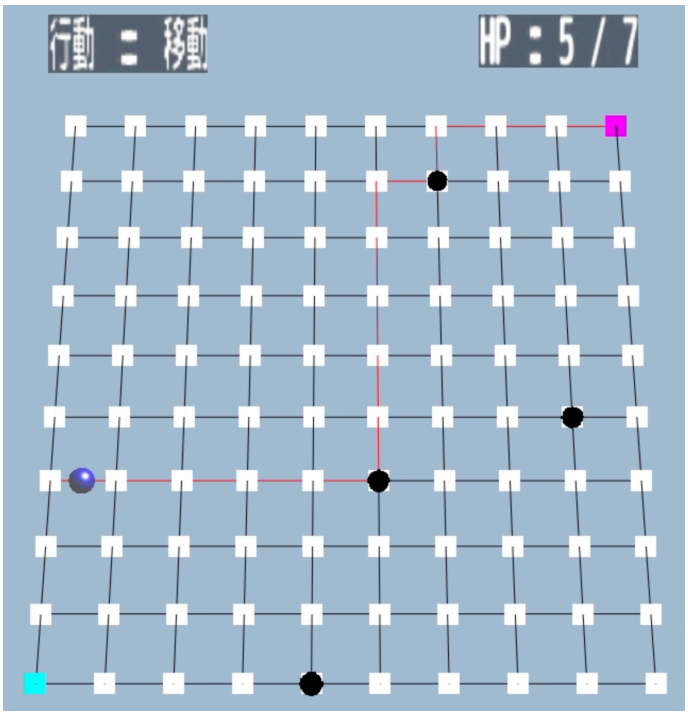

図 3.29　敵に攻撃した後に回復を行わず、次の地点に移動する様子.

次の攻撃対象であるマップ中央の敵に向かって移動中、新たに 1 体の敵が攻撃対象となる状況が発生した。この時、エージェントの体力値は 5 であり、倒す敵の対象の数は 3 体となるため、回復をしなければ、ゴール地点まで辿り着くことができなかった。そこで、回復を行うことが必要であると判断し、移動途中で回復を行った。図 3.30 に、移動途中で回復を行った様子を示す。

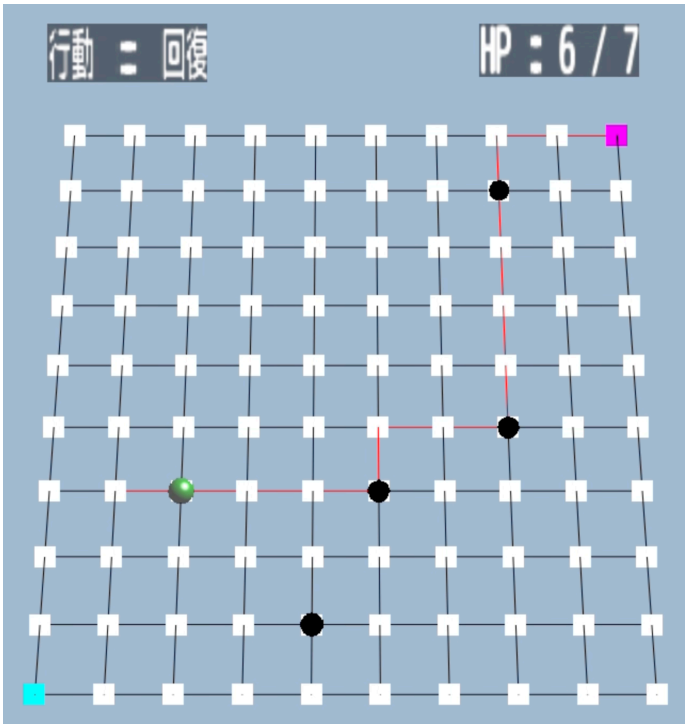

図 3.30　移動途中で回復を行う様子.

しかし、新たに攻撃対象となった敵はエージェントの行動経路から外れた地点に移動するため、対象となる敵の数は 2 体に戻り、追加の回復を行う必要は無くなった。その後は、マップ中央の敵と上部の敵を倒し、ゴール地点まで到達した。ゴール地点に到達した時の体力値は、2 となった。図 3.31 に、ゴール地点に到達した時の様子を示す。結果的に、ゴール地点到着時の体力値が 2 となっていて、移動途中に回復を行う必要はなかったが、攻撃対象が 3 体に対して体力値が 5 の状況では、回復を行うことが必要と判断した。本手法では、未来の行動や状況を予測した行動算出するのではなく、その状況で制約条件を満たす行動経路を算出するということが目的となるため、今回のような結果となった。

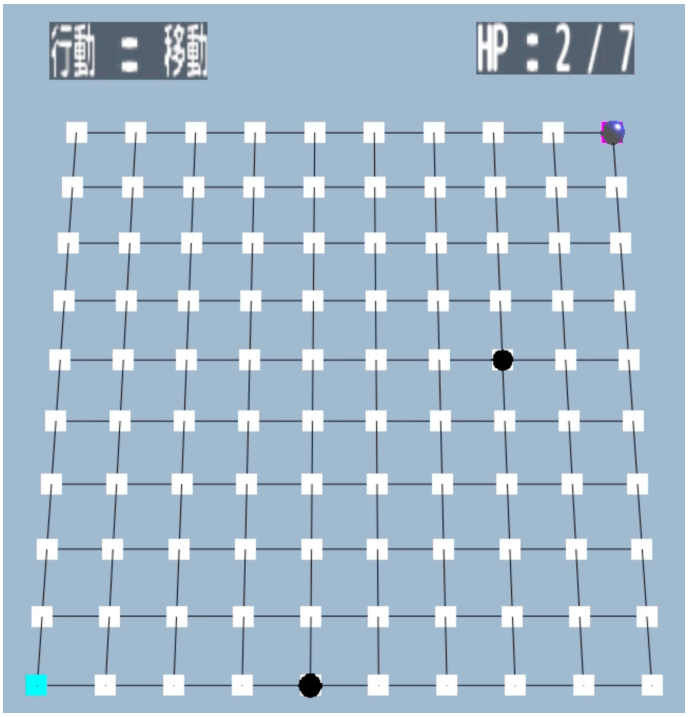

図 3.31　移動途中で回復を行う様子.

3.6.4　体力値上限を 7 とする通過制限なし探索での検証

3.6.2 項の体力値上限を 5 とする通過制限なし探索の検証に対して、体力値上限を 7 として検証を行った。この時、λ の値は 6 とした。倒す敵の順番を決める方法についても同様の式を用いて検証を行った。図 3.28 に、体力値上限を 7 とする通過制限なし探索での実行画面を示す。

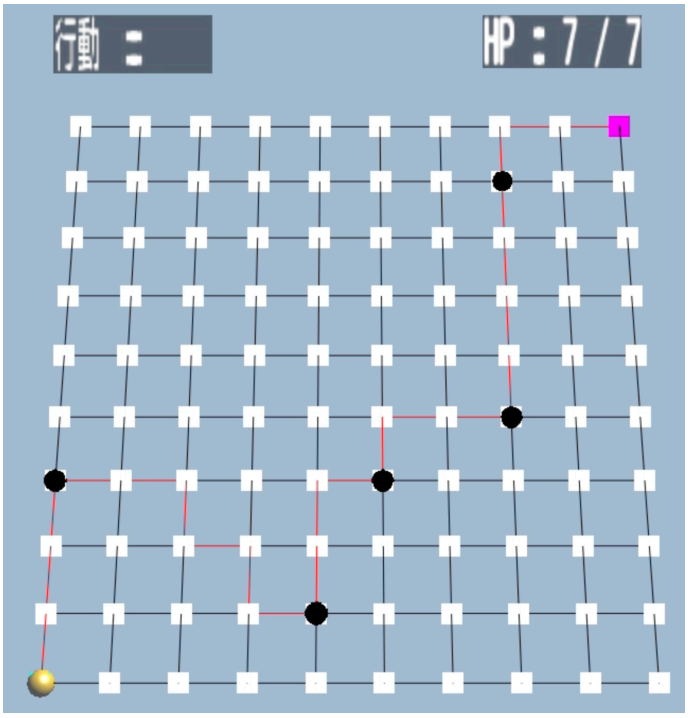

図 3.32　体力値上限を 7 とする通過制限なし探索での実行画面.

1 体目と 2 体目の敵を倒した後の行動は、3.6.2 項の体力値上限を 5 とする通過制限なし探索の検証と同様の行動経路が算出され、攻撃した地点で回復を行い、別地点に移動し、回復を行うという結果になった。図 3.33 に、2 体目の敵を倒し、別地点に移動して体力値を 7 まで回復した様子を示す。

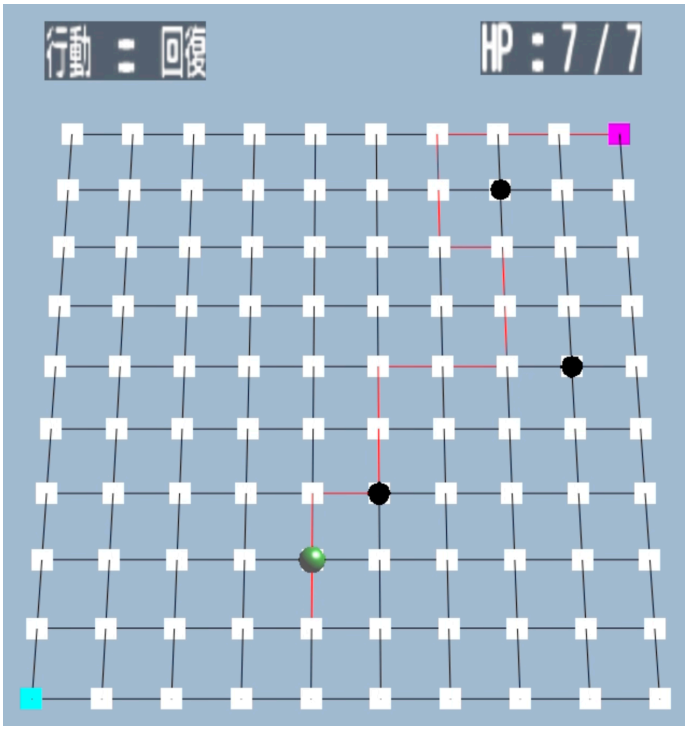

図 3.33　2 体目の敵を倒し、別地点で回復をして体力値を 7 まで回復した様子.

3 体目の敵となるマップ中央の敵に攻撃する際は、攻撃後に回復を行わず、次の敵に向かって移動した。図 3.34 に、攻撃後に回復を行わずに、次の地点に向かって移動する様子を示す。

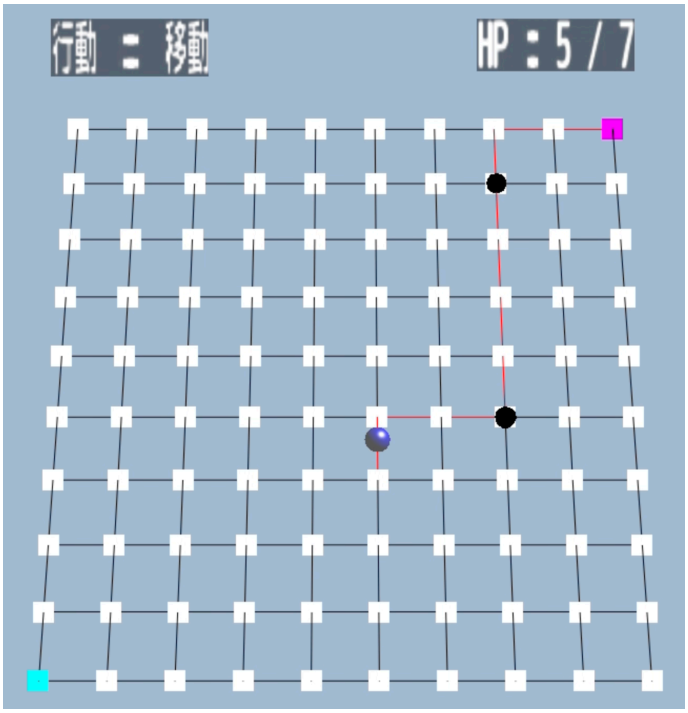

図 3.34　3 体目の敵を倒し、回復を行わずに、次の地点に向かって移動する様子.

3 体目の敵を倒して次の敵に向かう行動経路に関しては、3.6.2 項の体力値上限を 5 とする通過制限なし探索の検証と同様の行動経路が算出された。ゴール地点に到達した時の体力値は、1 となった。

3.6.5　初期体力値が減少した状態における通過制限あり探索での検証

本項では、敵を倒すことを優先事項とする状態で、体力値の初期状態を 5 ではなく、1 の状態から行動経路探索を行った。敵の位置や移動順路などは同条件で検証を行った。図 3.35 に、初期体力値が減少した状態での敵を倒すことを優先事項とする通過制限あり探索による結果の様子を示す。

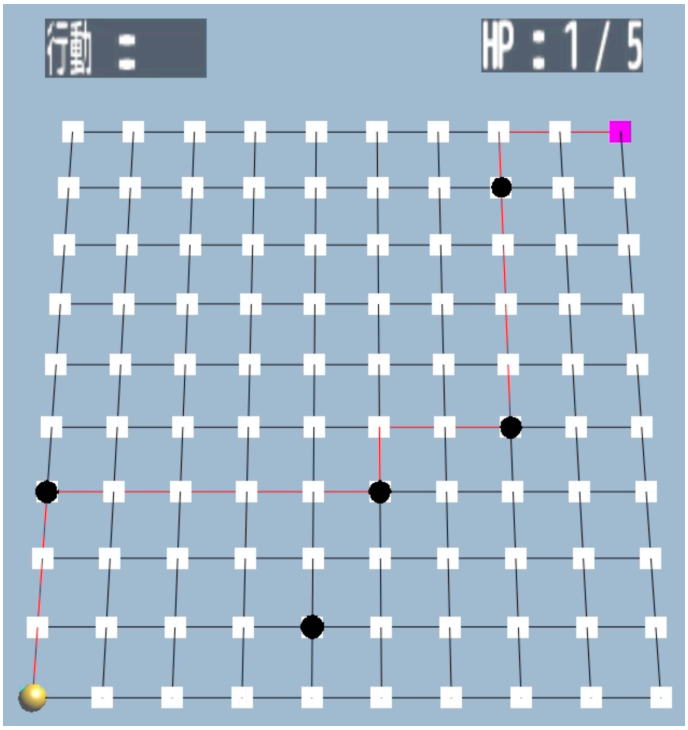

図 3.35　体力値が減少した状態での通過制限あり探索の実行画面.

この状態から、エージェントは 2 地点にわたり回復を行う。図 3.36 に、スタート地点から 2 つ上の地点までで、回復を 2 度行い、エージェントの体力値が 3 まで回復した様子を示す。

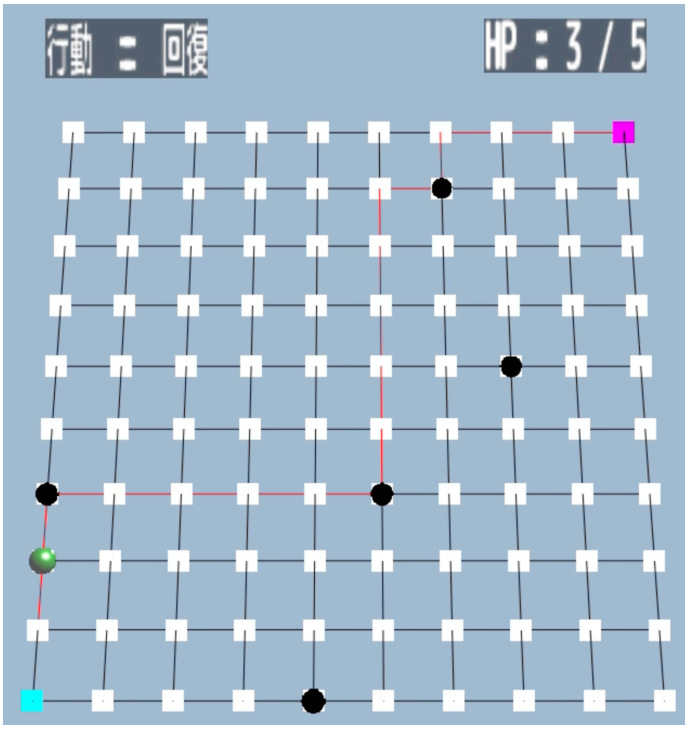

図 3.36　2 地点で回復を行った様子.

図 3.37 に、体力値が 3 の状態で、敵に攻撃をし、同地点で回復を行った様子を示す。

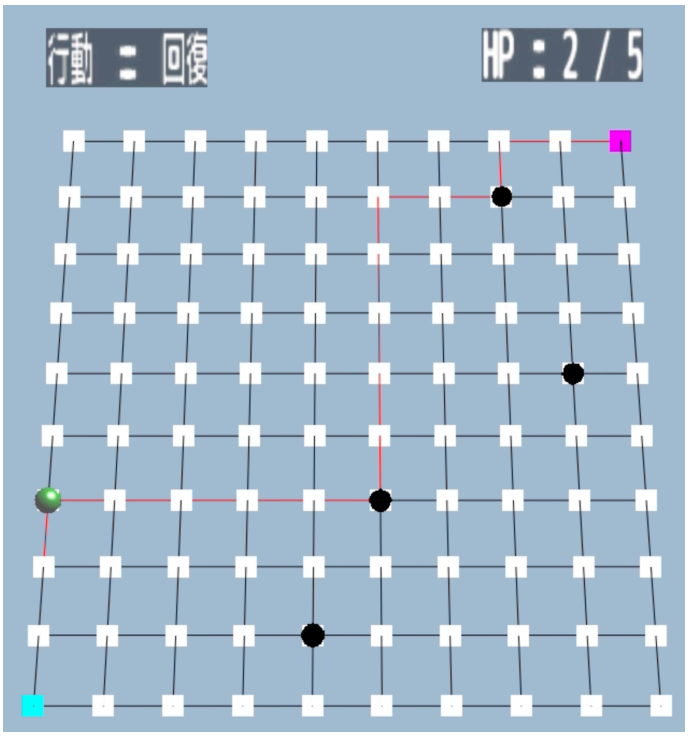

図 3.37　敵に攻撃をし、同地点で回復をする様子.

この後の行動経路は、マップ中央の敵を倒すまでに体力値を 4 まで回復し、敵を倒した後に 1 度回復を行った。図 3.38 は、マップ中央の敵を倒すまでに体力値を 4 まで回復した様子を、図 3.39 に、攻撃したのちに次の地点で回復を行う様子を示す。図 3.39 の地点にいる段階で回復をして、体力値を 3 にすることで、マップ上部にいる敵に攻撃しても体力値が 0 になることなくゴール地点に辿り着くことができた。最後にマップ上部にいる敵を倒してゴール地点に到達した。この時の体力値は、1 となった。

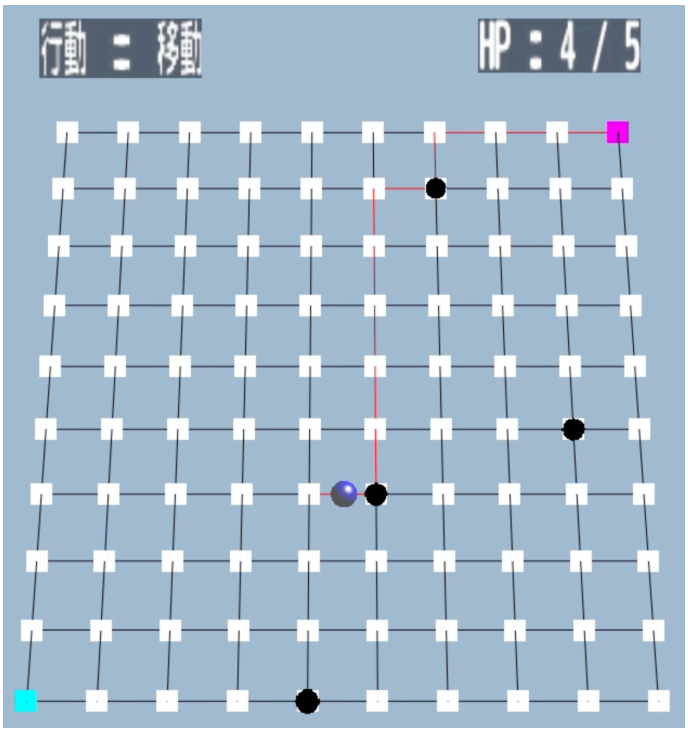

図 3.38　体力値を 4 まで回復した状態で敵の地点に向った様子.

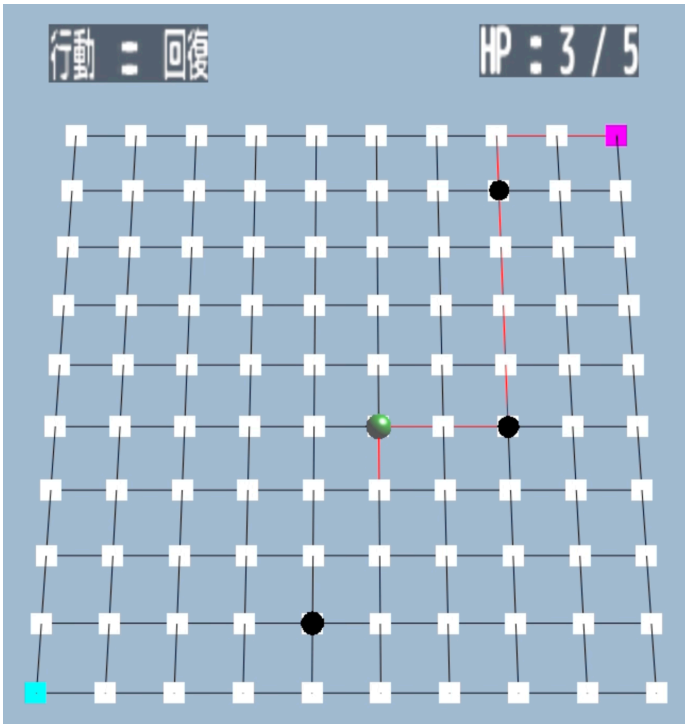

図 3.39　敵に攻撃したのちに、次の地点で回復を行った様子.

3.6.6　初期体力値が減少した状態における通過制限なし探索での検証

本項は、3.6.5 項と同様に敵を倒すことを優先事項とする状態で、体力値の初期状態を 5 ではなく、1 の状態から行動経路探索を行った。敵の位置や移動順路などは同条件で検証を行った。図 3.40 に、初期体力値が減少した状態での敵を倒すことを優先事項とする通過制限なし探索による結果の様子を示す。

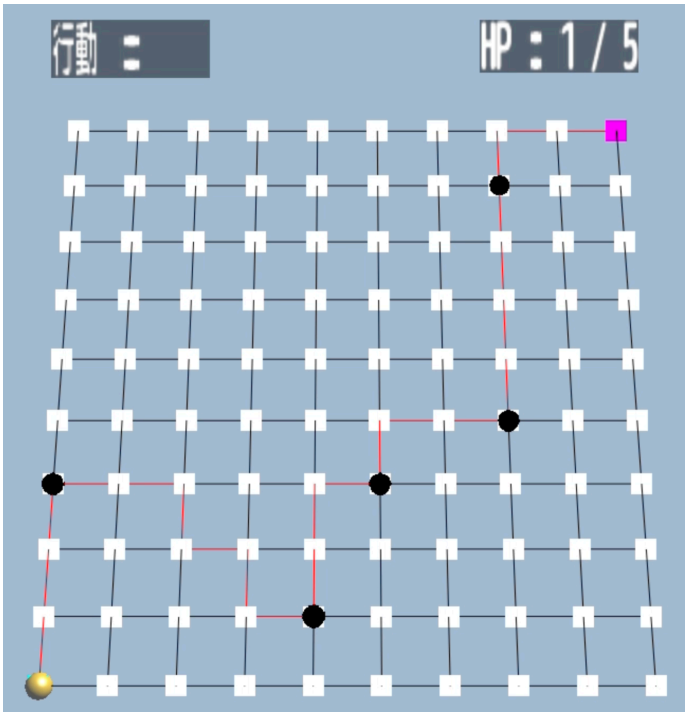

図 3.40　体力値が減少した状態での通過制限なし探索の実行画面.

通過制限あり探索と同様に、最初の敵を倒すまでに 2 度回復を行い、体力値を 3 の状態にして攻撃を行った。図 3.36 は、スタート地点から敵の地点に移動する際に、2 地点連続で回復を行い、体力値を 3 の状態にして敵地点に向かうエージェントの様子を示す。

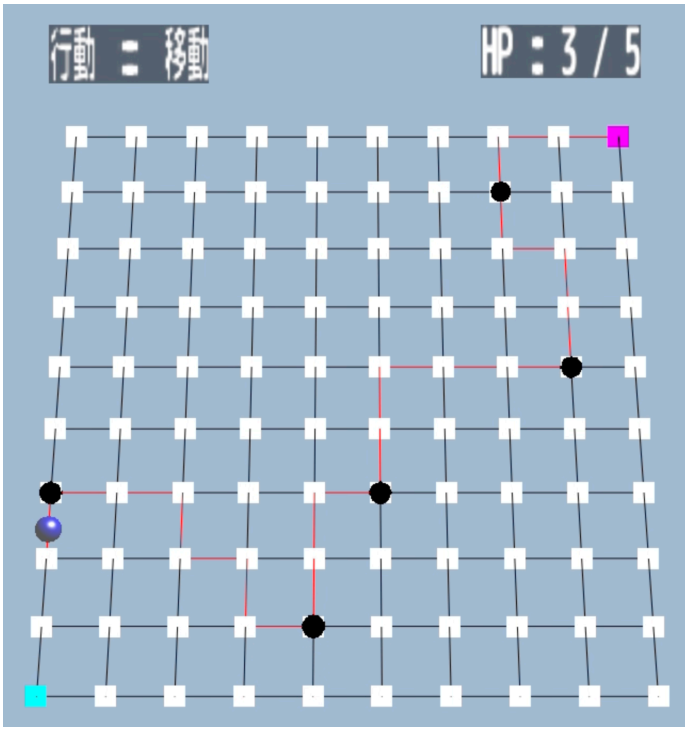

図 3.41　回復を 2 地点連続で行い、敵の地点に移動した様子.

その後、敵を攻撃し、同地点で回復を行ってから、3 地点続けて回復を行い、体力値を 5 の状態にして次の敵に向かって移動した。図 3.37 に、3 地点連続で回復を行い、体力値を 5 まで回復した様子を示す。

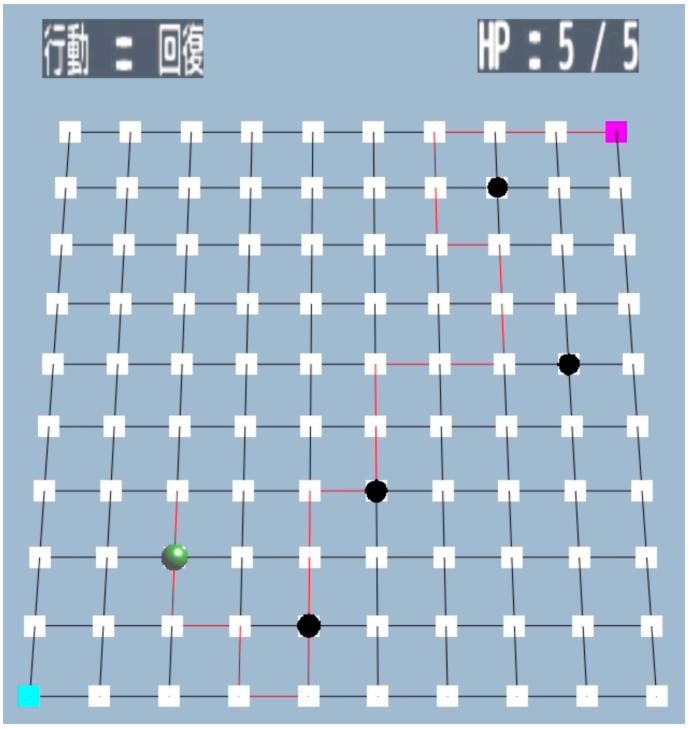

図 3.42　回復を 3 地点連続で行い、体力値を 5 まで回復した様子.

その後の行動経路に関しては、3.6.2 項と同様の手順で、行動していき、3 体目までの敵を倒した後は攻撃をして 2 地点連続で回復をするという流れになった。最終的に、体力値が 1 の状態でゴール地点に到達した。

3.7　最短経路を優先事項とする検証結果

本節では、最短経路を優先事項とした際の通過制限あり探索と通過制限なし探索の検証結果について説明する。最短経路を優先事項とした際の 2 つの手法での結果に違いはないため、まとめて紹介する。図 3.43 は、最短経路を優先事項とした際の実行画面を示す。

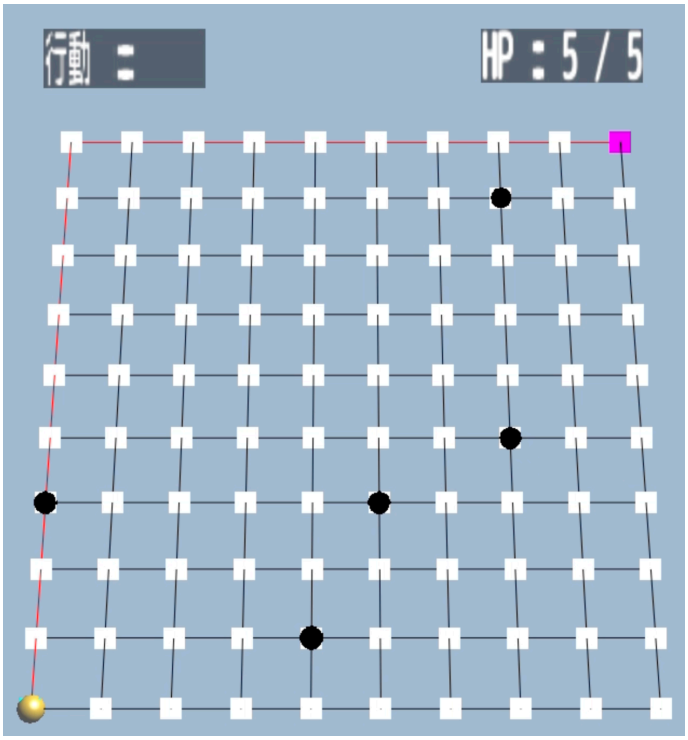

図 3.43　最短経路を優先事項とした際の実行画面.

敵の配置やエージェントの体力値の状態は、3.6.1 項の条件と同様のものとした。エージェントは、上方向に移動をしていき、敵のいる地点まで移動しても、攻撃することなくその次の地点に移動した。図 3.44 に敵に攻撃することなく次の地点に移動する様子を示す。

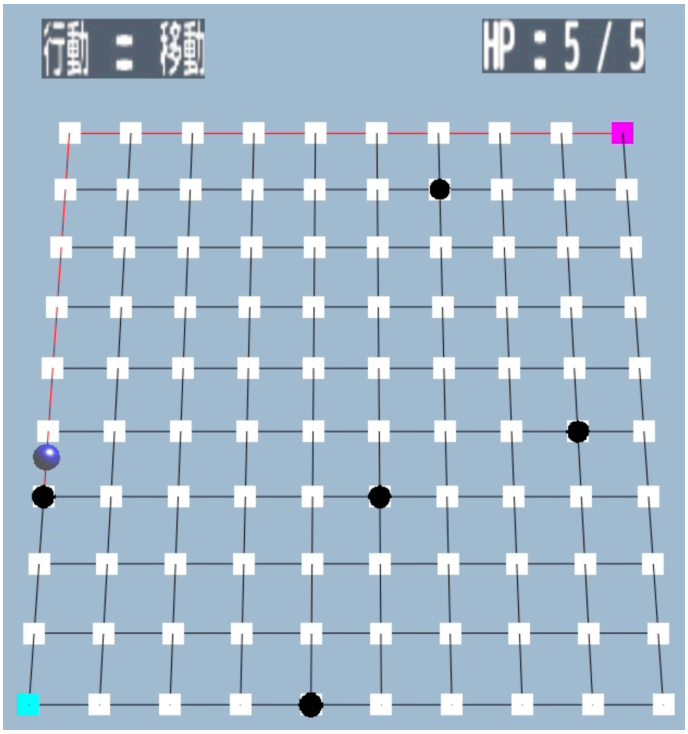

図 3.44　敵に攻撃することなく次の地点に移動する様子.

この後、算出した経路をもとに移動をしていき、ゴール地点に到達した。優先事項を変えることで、それぞれの条件に合った行動経路を算出することが可能であることがわかった。

3.8　処理時間の比較

表 3.2 に、2 種類の手法での処理時間を示す。いづれの検証条件でもスタート地点からゴール地点にかけての探索が処理時間のかかるものであるため、プログラム実行時の探索における処理時間を記載する。

通過制限あり探索では、探索に制限があり、探索量が少ないため、探索時間が短く、状況による差異は見られなかった。しかし、通過制限なし探索では、探索に制限がないため、探索する状況によって処理時間に差異が見られた。特に、体力値上限が 7 の時の検証では、記録しておく候補が体力値上限が 5 の検証と比べて増加しているため、処理時間が大幅にかかっていることがわかった。一般的なゲームでは、60fps つまり、1 秒間に 60 フレームを描画しており、1 フレーム

あたり 167ms で全ての処理を行う必要がある。167ms で描画を含む全ての処理を行う必要があるため、そのうち AI の処理が占める割合が多いほど、処理負荷が高くなり、他の処理にも影響する。通過制限あり探索の結果のように、探索状況による処理時間の差異がなく、10ms 程度であれば、AI にかける時間が少ないため、処理負荷が低い。しかし、通過制限なし探索の処理は最大で 129ms とほとんどの時間を AI に割く必要があるため、60 フレームでの描画は現実的ではなくなってしまう。通過制限なし探索では、探索状況による処理時間の差異が顕著であり、いくつかのケースで問題となることがわかった。そのため、毎秒処理を行うのではなく、数秒に 1 回の処理にするなどの工夫が必要となる。

表 3.2　処理時間の比較 (ms)

検証状況	通過制限あり探索	通過制限なし探索
体力値上限 5	10	59
体力値上限 7	11	129
体力値減少	11	82
最短経路	9	17

3.9　考察

本研究では、エージェントの各地点における行動を経路探索におけるノードとして扱うことで、複数の制約条件を考慮した行動経路の算出を行うことを目的として、最短経路を優先事項とした場合と敵を倒すことを優先事項とした場合での検証を行った。最短経路を優先事項とした場合はどちらの手法でも同様の結果が得られた。敵を倒すことを優先事項とした際の通過制限あり探索では、探索の制限があり、同地点を 1 度のみしか通過できないため、3 体の敵を倒すという結果になった。対して、通過制限なし探索では、迂回するケースにも対応して 5 体全ての敵を倒す行動経路を算出することができた。しかし、通過制限あり探索の方が処理時間が高速であるため、使用場面に制限はあるものの、処理時間を減らしたい場合は、通過制限あり探索を用いることで、探索処理を減らせることがわかった。そのため、通過制限あり探索と通過制限なし探索はそれぞれ

に利点があるため、使用場面に応じて手法を使い分ける必要がある。

実際のゲームでは、グラフに制限があるという場合が少ないため、通過制限あり探索が適用可能となる適用場面を考察していく必要がある。現状、考えられる適用場面としては、スゴロクゲームのような移動する向きがある程度決まっているものでの使用である。具体的なゲーム作品として、「マリオパーティ」や「桃太郎電鉄」が挙げられる。マリオパーティでは、スターと呼ばれるものを集めること目的とするゲームである。スターを獲得するためには、コインを規定数集め、他のプレイヤーよりも早くスターのある地点に移動する必要がある。プレイヤーの代わりを務めるNPC は、コインを集めつつ、スターのある地点に移動し、他のプレイヤーよりも多くのスターを集める必要がある。これらの条件を制約条件とすることで、通過制限あり探索での探索が実現できると考えた。同様に、桃太郎電鉄も他のプレイヤーよりも早く目的地点となる駅に到着することを目指す。しかし、桃太郎電鉄では、最終的な総資産が他のプレイヤーよりも多いことが勝利条件となる。そのため、停車した駅で買い物をしたり、賞金のもらえる地点に移動するなどして、最終的な総資産のトップになることを目的としている。このようなスゴロクゲームでは、行動を行う方向が決まっているため、通過制限あり探索が適用できる場面であると考える。

通過制限なし探索は、グラフに制限がないため、探索状況によってはゲーム全般的に応用も考えられるが、処理速度に問題があるため、処理速度の高速化の工夫が必要である。処理速度の問題が出る原因として、ノードが通常の移動経路の探索に比べ、3 倍となり、エッジ数も増加しており、探索処理が多くなっていることが挙げられる。今後は、行動ノードの管理やエッジ数を削減するという改善が必要となる。今回の探索では、ダイクストラ法をベースとして探索を行っていたため、A*法のように、探索処理を削減するような経路探索手法での改良を行うことで、処理の削減が実現できると考えている。A*法を用いる際に、ヒューリスティック関数として考えられるのは、対象とする経由地点までの距離などである。その他の工夫点としては、ゲームにおける経路探索でも用いられる階層化による探索処理の削減である。階層化による探索は、経路探索を行う際に、マップを単純な構造にしたグラフで探索を行うことで経路の道筋を決定し、それをもと

により複雑なグラフで探索を行うことで、処理削減を図るものである。これら以外にも経路探索における処理削減の手法は複数存在する。このような改良を行うことで、現在の手法よりも高速に処理を行うことが期待できる。

今回の検証では、コストの値が図 3.1 に示した値で固定となっていた。そのため、行動の流れがパターン化しているように感じられた。今後は、コストの値を地点ごとに変化させることで、行動にどのような変化が起きるのかを検証する必要がある。具体的には、敵の強さに合わせて攻撃ポイントの値を変更したり、地点間の移動時間を変化させたりすることである。また、エージェントの行動が 3 種類と少ない環境で行っていたため、「アイテムを使う」、「その場で待機」などの新たな行動にも対応するように手法を拡張していくことで、ゲームでの応用が現実的になると考えている。また、今回の検証では、「攻撃」と単純なノードとしていたが、実際のゲームでは、攻撃の種類が多数存在する。例としては、物理攻撃や魔法攻撃などである。これは、攻撃だけでなく、その他の行動でも同じことが言える。このようにさらに細かな行動が存在する場合は、上記でも紹介した階層化を行うことで、処理削減ができると考えている。行動を全てノードにしてしまうと、ノード数やエッジ数が膨大な数になってしまい、管理が困難になってしまうため、抽象的な行動グラフから探索をしていくことで対応する。

現状、1 体のエージェントのみに本手法を適用した検証しか行えなかった。今後は、複数のエージェントが連携できるようにすることで、適用場面を増やすことができるようになることが考えられる。本手法は、複数のエージェントの連携には対応していないため、マルチエージェントの経路探索手法を適用することを検討する必要がある。通常の経路探索同様、マルチエージェントの経路探索も多くの研究が行われている。複数のエージェントが連携できれば、1 体のエージェントで探索するよりも効率よく、制約条件を満たす行動経路を算出できるようなる可能性もある。また、本手法は、プレイヤーキャラクターとエージェントの連携にはあまり適さないと考えている。なぜなら、プレイヤーキャラクターはエージェントが想定している行動を必ずしも取るとは限らないからである。例えば、何らかのミッションを実行中にプレイヤーキャラクターが長時間

静止し続けることやミッションとは全く関係のない地点に移動することなどである。このような状況でも、本手法を適用したエージェントであれば、プレイヤーキャラクターとの連携がなくてもミッションをクリアしてしまう。そうなってしまうと、ゲームとしての楽しみがなくなってしまうため、プレイヤーキャラクターとの連携をするのは、本手法は適さないと判断した。

今後の展望として、2 つ挙げる。第一に、より効率の良く経由地点の訪問する順番を決める手法の提案である。今回は、比較的容易な順番の決め方をしていたため、今後は巡回セールスマン問題などの手法を応用することが望ましいと考えている。第二に、行動の種類を追加することで、エージェントの行動をより柔軟に行えるようにすることである。単純にノードを追加していくと先述の通り、処理時間が増加するだけとなってしまうため、階層化して処理を行うことやノードやエッジのデータの持たせ方を改善することが必要である。また、今回は、最大で 100 地点での検証しか行えていないため、処理の高速化を実現し、より多くの地点数で処理速度の削減ができれば、ゲームでの使用が現実的になる。

第 4 章

まとめ

これまでの NPC の行動決定の方法では、大まかな行動でしか行動を管理することができず、長期的な目標を達成することと複数の制約条件を考慮した行動を算出することができなかった。本研究は、ゲームのマップ上における各地点での NPC の行動を経路探索におけるノードとして扱うことで、ゴールベース AI の特徴でもある長期的な目標の達成のために必要となる行動手順を算出する。その際、制約付き最短経路問題に基づき、複数の制約条件を考慮した行動経路を算出する手法を提案した。本手法は、ゲーム AI における NPC の行動決定を行うキャラクター AI と、移動経路を決定するナビゲーション AI の 2 つの役割を合わせたものになる。マップ上の各地点に複数の行動ノードを設定することで、経路探索と同様の手順で、行動経路を算出する。各エッジには重みが設定されており、その重みの総和をもとに制約条件を考慮した経路を各地点において記録することで、行動経路を算出した。実験に使用したグラフの規模は、小規模であったものの、行動をノードとして扱い、複数の制約条件を考慮した行動経路を求めることができた。また、通過制限あり探索と通過制限なし探索の検証をしたことで、以下のことがわかった。通過制限あり探索では、グラフの制限範囲内で、制約条件を満たす行動経路を算出することができ、通過制限なし探索に比べて高速に処理を行うことができた。通過制限なし探索では、処理が通過制限あり探索に比べて遅いものの、グラフの制限なく、同地点を複数回通過するケースに対応し、制約条件を満たす行動経路を算出することができた。

しかし、本手法では、通常の経路探索における地点間のエッジ以外に、各地点にある行動ノード同士にもエッジがあるため、エッジの総数が多くなり、エッジの管理が困難であることが問題として挙げられる。通過制限あり探索では、スゴロクゲーム以外での適用場面の考察を行い、通過制限なし探索では、高速化を実現するために他の経路探索手法を適用するなどの発展させていくことが必要である。また、ゲームのようにリアルタイムでの使用を想定した場合、以下の 2 点を実現する必要がある。第一に、行動を階層化することで、より多くの行動決定ができるようにすることである。第二に、処理負荷を軽減するために、使用するエージェントを絞って活用する、または経路探索の高速化を行うことである。

謝辞

本研究、本執筆を行うにあたり、数多くのご指導をしていただいた渡辺先生、阿部先生、そして共に研究に励み、支え合った研究室のメンバーに感謝申し上げます。渡辺先生や阿部先生には、自分では気づけない問題や対策などをご指導していただき、時には研究の軌道修正を行うなど、研究において何度も助けていただくことがありました。本研究と同時期に執筆をしていた、学部時代の研究の論文誌投稿の際にも大変お世話になりました。先生方のおかげで、論文誌に掲載することができたので、本当に感謝しております。この実績は今後の人生において大きな影響を与えるものだと実感しており、再度研究活動を行う際の糧にしていきたいと思います。

また、中間審査会や最終審査会で審査をしていただいた竹島先生、寺澤先生、加藤先生にも大変お世話になりました。自身の研究を上手く伝えることができず、ご迷惑をかけてしまうこともありましたが、先生方は研究の意図を汲み取り、研究にとって有意義なご指摘をしていただきました。

加えて、学会で貴重な意見やアドバイスをしていただいた先生方や企業の方々にも深く感謝しています。緊張で上手く発表や質疑応答ができない場面もありましたが、私の発表を真剣に聴講していただきありがとうございました。

さらに、研究活動を通してたくさんの経験をできたことに感謝します。学会では普段の生活では体験できないような貴重な経験をさせていただき、たくさんの思い出を作ることができました。そして、自分の専門分野だけでなく、幅広い分野の知見を得ることができました。

最後に、私を支えてくれた皆様に心より感謝を申し上げます。本研究は私に関わっていただいた皆様のおかげで成り立っていると実感しております。時には研究成果が出せず、挫けそうになる場面も多々ありましたが、親身に寄り添っていただいた皆様のおかげでやり遂げることができました。そして、私の学業を支えてくれた両親にも改めて感謝を申し上げます。

皆様、本当にありがとうございました。

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] 森川幸人. テレビゲームへの人工知能技術の利用 (ai 技術の産業応用〔第 7 回〕). 人工知能, Vol. 14, No. 2, pp. 214–218, 1999.

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

[4] 三宅陽一郎. 大規模 デジタルゲームにおける人工知能の一般的体系と実装-FINAL FANTASY XV の実例を基に. 人工知能学会論文誌, Vol. 35, No. 2, pp. B–J64_1, 2020.

[5] 2016-2019 © SQUARE ENIX CO., LTD. FINAL FANTASY XV（ファイナルファンタジー 15）—SQUARE ENIX. http://www.jp.square-enix.com/ff15/. 参照: 2022.11.10.

[6] © 2016 Take-Two Interactive Software, Inc. GTA Online - Rockstar Games. https://www.rockstargames.com/jp/gta-online. 参照: 2022.11.10.

[7] © 2020 KADOKAWA Game Linkage Inc. ゲーム ai 用語辞典. https://wiki.denfaminicogamer.jp/ai_wiki/. 参照: 2022.11.17.

[8] 森寅嘉, 角薫. 状況に応じた動作を制御するキャラクタ AI. 研究報告エンタテインメントコンピューティング (EC), Vol. 33, pp. 1–7, 2018.

[9] Meili Zhu and Lili Feng. Design and Implementation of NPC AI Based on Genetic Algorithm and BP Neural Network. *Proceedings of the 14th International Conference on Computer Modeling and Simulation*, No. 6, pp. 168–173, 2022.

[10] Boeda Gautier, 三宅陽一郎, 坂田新平, Martins Gustavo. キャラクター ai の成長と学習のアーキテクチャ. 人工知能学会全国大会論文集, Vol. JSAI2022, pp. 2O5GS501–2O5GS501, 2022.

[11] © 2021-2022 Inworld AI. Inworld ai – create ai characters and ask them anything. https://www.inworld.ai/. 参照: 2022.11.17.

[12] © 2022NTT レゾナント株式会社. Ai suite(エーアイスイート). https://aisuite.jp/. 参

照: 2022.11.17.

[13] © 2022 CESA All rights reserved. Cedec2022. https://cedec.cesa.or.jp/2022/session/detail/105. 参照: 2022.11.21.

[14] 張輝陽, 星野准一. プレイヤ行動の模倣に基づく AI キャラクタ行動ルールの自動生成. Technical report, 2014.

[15] 星野准一, 田中彰人, 濱名克季. 模倣学習により成長する格闘ゲームキャラクタ. 情報処理学会論文誌, Vol. 49, No. 7, pp. 2539–2548, 2008.

[16] 長友結希, 三宅陽一郎. 強化学習によるエージェントの戦術獲得の分析. ゲームプログラミングワークショップ 2021 論文集, No. 2021, pp. 106–110, 2021.

[17] ゲーム AI と深層学習-ニューロ進化と人間性-. オーム社, 2018.

[18] 三宅陽一郎, 水野勇太, 里井大輝. 「メタ ai」と「ai director」の歴史的発展. デジタルゲーム学研究, Vol. 13, No. 2, pp. 1–12, 2020.

[19] © 2011-2022 Tokyo Electron Limited. All Rights Reserved. 自分で考え行動するゲームキャラクター 進化するゲーム世界と現実世界への応用 (3/4) — telescope magazine. https://www.tel.co.jp/museum/magazine/021/interview02/03.html. 参照: 2022.11.21.

[20] 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. *CEDEC2006*, 2006.

[21] Jeff Orkin. Three States and a Plan: The A.I. of F.E.A.R. 2006.

[22] Restuadi Studiawan, Mochamad Hariadi, and Surya Sumpeno. Tcatical planning in space game using goal-oriented action planning. *JAREE (Journal on Advanced Researchin Electrical Engineering)*, Vol. 2, No. 1, pp. 5–9, 2018.

[23] 古川真帆, 阿部雅樹, 渡辺大地. 長期目標達成を考慮したユーティリティベース AI に関する研究. 芸術科学会論文誌, Vol. 20, No. 2, pp. 139 – 148, 2021.

[24] H.C.Joksch. The shortest route problem with constraints. *Journal of Mathematical*

Analysis and Applications, Vol. 14, No. 2, pp. 191–197, 1966.

[25] 森畑明昌, 松崎公紀, 武市正人. 領域限定言語に基づく最適経路問合せ. 情報処理学会論文誌, Vol. 4, No. 2, pp. 116–133, 2010.

[26] R.J.Wilson. グラフ理論入門 (原書第 4 版). 近代科学社, 2001.

[27] 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.

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

[29] あたらしい数理最適化：Python 言語と Gurobi で解く. 近代科学社, 2012.

[30] 1997-2020 © Fine Kernel Project (fk@gamescience). Fine Kernel ToolKit Top Page. `https://gamescience.jp/FK/`. 参照: 2022.3.15.

発表業績

論文誌掲載 (査読付き)

1. 吉田涼真, 阿部雅樹, 渡辺大地, 事前探索データを用いた経路探索手法, 芸術科学会論文誌,Vol.21,No.2,pp.111-122,2022.

口頭発表 (査読付き)

1. 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,NICOGRAPH2021(フルペーパー採録),2021.
2. 吉田涼真, 阿部雅樹, 渡辺大地,NPC の行動の最善手探索に関する研究,NICOGRAPH2022(ショートペーパー採録),2022.

口頭発表 (査読無し)

1. 吉田涼真, 阿部雅樹, 渡辺大地,NPC の行動の最善手探索に関する研究, 芸術科学会東北支部研究会 (ショートペーパー採録),2022.
2. 吉田涼真, 阿部雅樹, 渡辺大地,NPC の行動の最善手探索に関する研究, 情報処理学会 (フルペーパー採録),2023.

ポスター発表

1. 吉田涼真, 渡辺大地,D*Lite を応用した新しい道ができたときの動的経路探索,NICOGRAPH2020,2020.
2. 吉田涼真, 阿部雅樹, 渡辺大地,NPC の行動の最善手探索に関する研究, 映像表現・芸術科学フォーラム 2022,2022.

受賞歴

1. NICOGRAPH2022 優秀論文賞 (ショートペーパー), 論文題目「NPC の行動の最善手探索に関する研究」,2022.