2019 年度　卒　業　論　文

コストマップの局所的な再設定による最短経路再探索手法の一般化についての研究

指導教員：渡辺 大地 准教授

メディア学部 ゲームサイエンス
学籍番号　M0116125
小坂 大樹

2020 年 2 月

2019 年度　　卒　業　論　文　概　要

論文題目

コストマップの局所的な再設定による
最短経路再探索手法の一般化についての研究

メディア学部 学籍番号：M0116125	氏名	小坂 大樹	指導教員	渡辺 大地 准教授

キーワード	経路探索, グラフ構造, コストマップ, アルゴリズム, 再探索

経路探索は、目標地点への移動経路を算出するものである。ゲーム AI やカーナビゲーションシステム、ロボット AI などに用いられている。経路探索手法にはダイクストラ法や A*アルゴリズム、D*アルゴリズムなど、多くの手法が用いられている。ダイクストラ法はノード間のコストから 2 点間の最適な経路を算出するものである。ダイクストラ法はゲーム AI や電車の乗換案内のシステムに用いられている。A*アルゴリズムはダイクストラ法をより効率的に処理できるよう改良したアルゴリズムである。ダイクストラ法や A*アルゴリズムは変化しないマップでは効果的であるが、マップが変化する場合には再探索が必要となる。D*アルゴリズムはマップ変化に対応しているが、新たな最短経路が出現する場合には対応していない。D*アルゴリズムが対応していないマップ変化に対応したのが、津川の提案した手法である。しかし、この手法は格子状のマップに対応しているが、汎用的なマップには対応していない。よって、津川の提案した手法を、汎用的なマップに対応させることが本研究の目的である。

提案手法では、一般的なグラフにおいて、エッジに割り当てたコストを元にしてノードのコストを更新し、スタートマップ・ゴールマップを作成する。そして、新たに経路が出現した場合はスタートマップの情報を基にゴールマップを局所的に更新する。最後に、スタートマップの情報と更新したゴールマップの情報を元に新たな最短経路を求める。提案手法の検証を行った結果、ノードが持つ接続関係が多い場合や少ない場合において、新たな最短経路を求めることができた。また、新たな最短経路の発生の有無を判別することができた。

目 次

図 目 次

第 1 章

はじめに

1.1　背景と目的

経路探索は、目標地点への移動経路を算出するものである。ゲーム AI[1][2] や、ロボット AI[3]、カーナビゲーションシステム [4][5]、災害時の経路シミュレーション [6][7] などの分野で用いられている。ゲームの分野において多く使用されているものでは、ナビゲーションメッシュ [8] や NavMesh[9] という手法がある。この手法では、ゲーム内の地形やオブジェクトを表すポリゴンメッシュ [10] を用いて経路探索を行う。カーナビゲーションシステムやロボット AI などの分野では、ダイクストラ法 [11] というアルゴリズムが用いられている。ダイクストラ法は、ノード間におけるコストから 2 点間の最適な経路を算出するものである。ダイクストラ法は乗換案内 [12][13] のシステムにも用いられており、交通費や所要時間をコストとすることで経路を探索している。図 1.1 はダイクストラ法の一般的な模式図を示す。

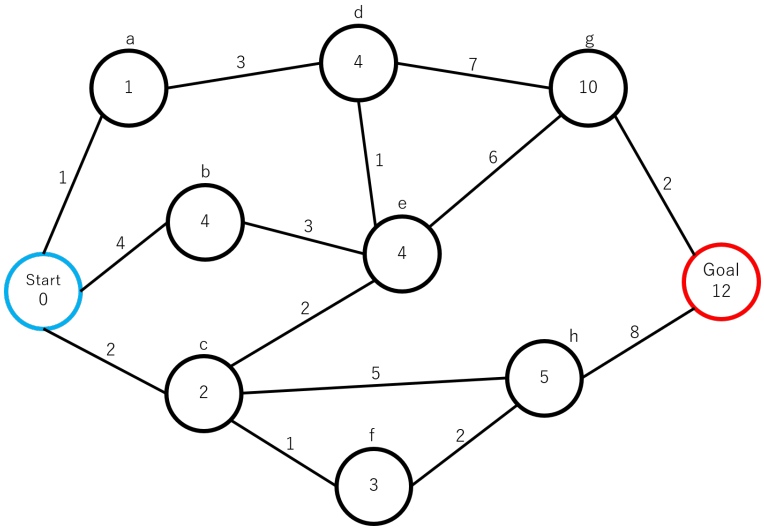

図 1.1　ダイクストラ法の一般的な模式図

円がノード、円と円を結ぶ直線がエッジ、エッジに付随して示している数値がコストに当たる。また、ダイクストラ法を改良したアルゴリズムとして、A*アルゴリズム [14] というアルゴリズムがある。

経路探索では目的地が変わることはよくあるが、経路探索を行うマップ自体は変化しないのが一般的である。マップが動的に変化する場合の探索では、変化のたびに再探索を行う必要がある。ダイクストラ法や A*アルゴリズムを用いた際も、マップ変化による経路変化のたびに再探索を行う必要がある。しかし、毎回マップすべてに対して再探索を行っていては処理時間が増加してしまい非効率的である。

動的な変化に対応するための経路探索の手法として、動的 A*(D*)[15] というアルゴリズムがある。動的 A*(D*) はダイクストラ法や A*アルゴリズムと比較すると動的な変化に対応したアルゴリズムである。しかし、この手法ではマップ変化により最短経路となる経路が塞がった場合には効果的に再探索を行えるが、マップ変化により新たな最短経路が出現した場合には冗長な処理が行われてしまう。そこで津川は新たな最短経路が出現した場合についての手法 [16] を提案した。この手法では、マップ変化により新たな最短経路が出現した場合に、少ない処理時間で再探索を

行うことが可能である。しかし、この手法では格子状のマップに特化したアルゴリズムとなっているため、図 1.1 のように一般的なグラフ構造をしたマップに対応していない。ゲームの分野においては、マップ変化により新たに経路が発生する状況が考えられる。また、近年のゲームにおいてはオープンワールドゲームの様な広いマップで、津川の提案した手法では対応できない汎用的なマップであることが多い。本研究の目的は、津川の提案した手法を元に、汎用的なマップにおいて新たな最短経路が出現した場合に、少ない処理時間で再探索を行う手法の提案である。

1.2 論文構成

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

第 2 章

先行研究

本章では、経路探索の手法について説明する。2.1 節ではグラフについて説明する。2.2 節では、探索アルゴリズムについて説明し、2.3 節では、現状の問題点について説明する。

2.1　グラフについて

経路探索ではグラフ理論におけるグラフ [17] を用いることがある。図 2.1 は一般的なグラフを示す。

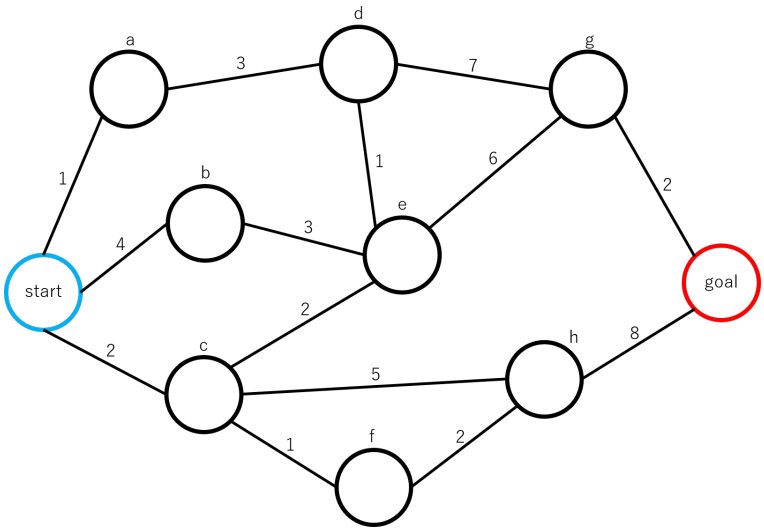

図 2.1　グラフ構造におけるグラフの一例

このグラフは、ノードとエッジというもので構成されており、ノードは節点または頂点、エッジは枝、辺ともいう。図 2.1 において、円がノード、直線がエッジとなる。2 つのノードがエッジで繋がれている状態を接続関係を持つといい、接続関係を持つノード間は移動可能とする。反対に、2 つのノードがエッジで繋がれていない状態を接続関係を持たないといい、接続関係を持たないノード間は移動不可とする。エッジにはそれぞれ数値が与えられており、エッジで繋がれている 2 つのノード間を移動する際のコストとなる。本研究では、スタート地点となるノードをスタートノード、ゴール地点となるノードをゴールノードとし、それ以外のノードを通常ノードとする。図 2.1 では通常ノードの判別のため記号を振っており、a から h までの通常ノードがある。また、グラフにおいてノードやエッジにコストを設定したものを本研究では、コストマップ [18][19] と定義する。

2.2 探索アルゴリズム

経路探索は、スタート地点からゴール地点までの経路をノードの接続関係やエッジのコストから求める。ここで求まる経路は、一般的に最もコストの小さい経路であることが多い。本節ではスタート地点からゴール地点までの最短経路を求める探索アルゴリズムを紹介する。

2.2.1 ダイクストラ法

グラフにおけるダイクストラ法を用いた経路探索について説明する。はじめに、グラフのエッジにコストを割り当てる。そして、スタートノードのコストを 0、ゴールノードと通常ノードのコストを∞に設定する。実際は、ゴールノードと通常ノードの初期コストには 10000 などの大きめの数値を設定しておく。図 2.2 はコストを割り当てた様子を表すグラフである。ノード内側の数値は各ノードのコストを表している。また、エッジ上の数値は各エッジのコストを表している。ダイクストラ法ではスタートノードから各ノードへの経路を求める。そのため、スタートノード

以外の各ノードはスタートノードからの経路が確定したか判別する必要がある。

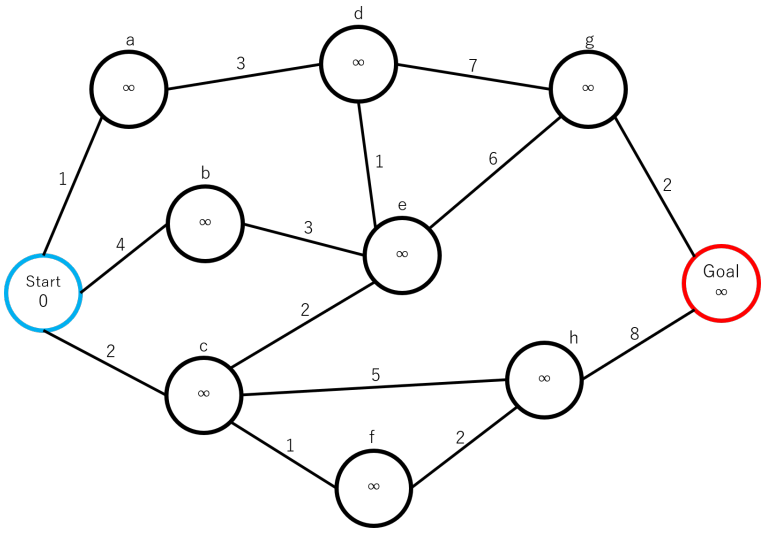

図 2.2　ダイクストラ法における初期コスト設定時の様子

このグラフにおける各ノードと各エッジのコストを元に各ノードのコスト更新判定を行う。あるノード α と接続関係にあるノード β の 2 つのノードにおいて、ノード α のコストと 2 つのノードを繋ぐエッジのコストを足した値とノード β のコストを比較する。そして、ノード α とエッジのコストを足した値がノード β のコストよりも小さい場合、足した値をノード β の新たなコストに更新する。大きい場合、コストの更新を行わない。スタートノードを除くその他全てのノードについて、初期コストの更新が終了する事で、スタートノードから各ノードへの経路が決定できる。

初めにスタートノードと接続関係のあるノードのコストを更新し、コストを更新したノードの内、最もコストの小さいノードを選択する。そして、スタートノードから選択したノードまでの経路が確定する。図 2.3 はスタートノードと接続関係にあるノードの内、最小コストとなるノードへの経路が確定した状態を表すグラフである。図 2.3 では、まずスタートノードと接続関係にあるノードを繋ぐエッジのコストを元に、ノード a, ノード b, ノード c を更新する。更新した後のノード a, ノード b, ノード c のコストはそれぞれ 1,4,2 となる。この中で最小のコストはノード a

の 1 であるため、スタートノードからノード a までの経路としてスタートノード、ノード a という経路が確定する。

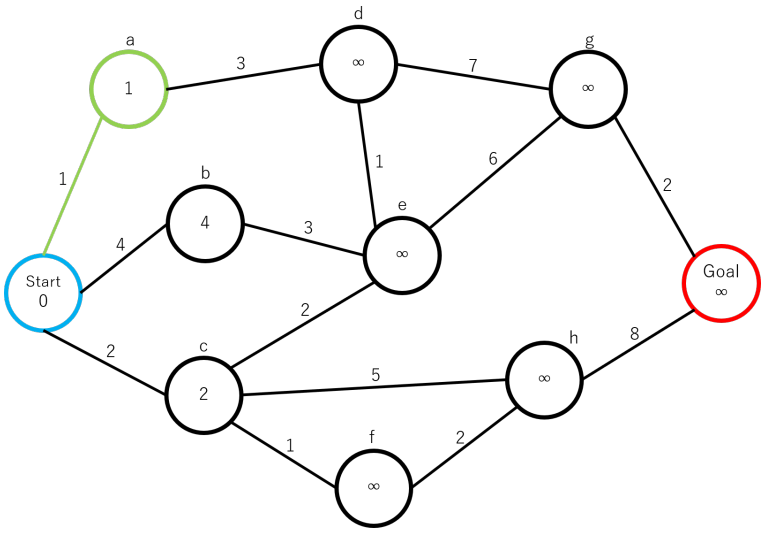

図 2.3　1 つ目のノードの経路決定

次に、スタートノードからの経路が確定したノードと接続関係にあるノードの内、経路が決定していないノードに対してコストの更新を行う。コストの更新を行った後、初めの手順においても未確定だったノードを含めて、スタートノードからの経路が確定していないノードの内、最もコストが小さいノードを選択する。先ほどと同様に、スタートノードから選択したノードまでの経路が確定する。上記の手順を繰り返し、スタートノードからゴールノードまでの最短経路を求める。図 2.4 は 2 つ目のノードの経路が確定した様子を示す。

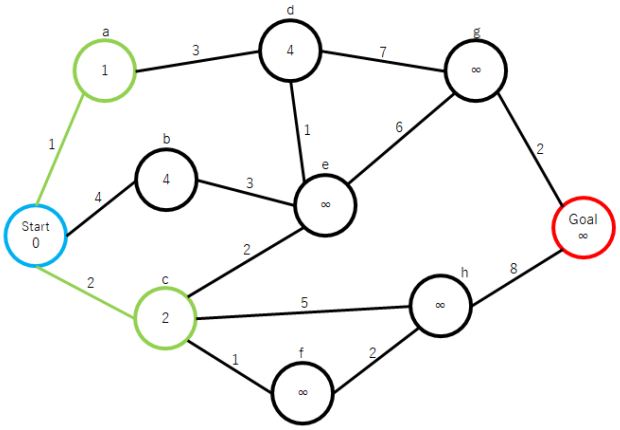

図 2.4　2 つ目のノードの経路決定

図 2.4 ではノード a までの経路が確定した為、ノード a と接続関係にあるノード d のコストを更新する。更新後、スタートノードからの経路が未確定なノード b、c、d のコストを比較し、最小となるノード c への経路が確定している。図 2.5 は経路がすべて求まった状態を表すグラフである。図 2.5 では、スタートノードから、ノード c、ノード e、ノード g、ゴールノードという、スタート地点からゴール地点までの最短経路が求まっている。

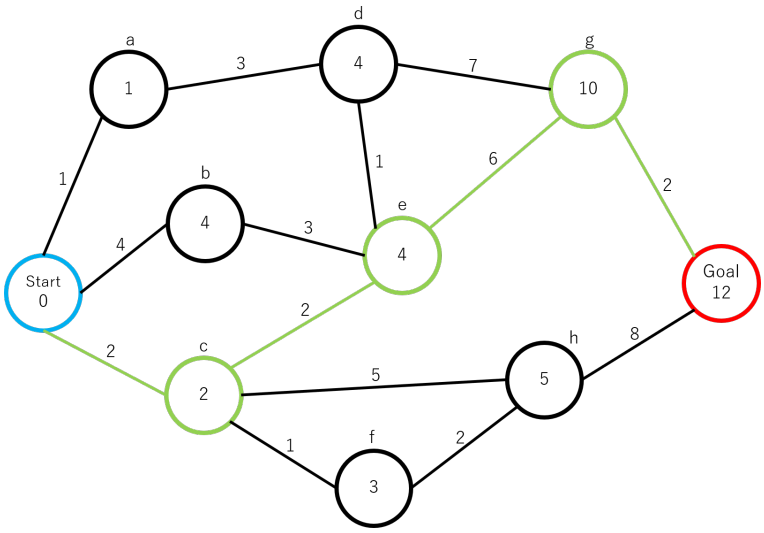

図 2.5　最終経路決定

2.2.2　A*アルゴリズム

A*アルゴリズムはダイクストラ法を改良したアルゴリズムである。基本的な流れはダイクストラ法と同様である。しかし、A*アルゴリズムはダイクストラ法で用いられたエッジのコストに加え、ヒューリスティックコスト [20] というものをあらかじめ設定する必要がある。ヒューリスティックコストとは、あらかじめ判明している情報などから設定しておくコストのことである。このヒューリスティックコストを追加することで、ダイクストラ法よりも効率的な経路探索が可能となっている。図 2.6 は、図 2.2 にヒューリスティックコストを追加したものである。図 2.6 では () 内の数値がヒューリスティックコストを表しており、ゴールノードからの接続数を設定している。

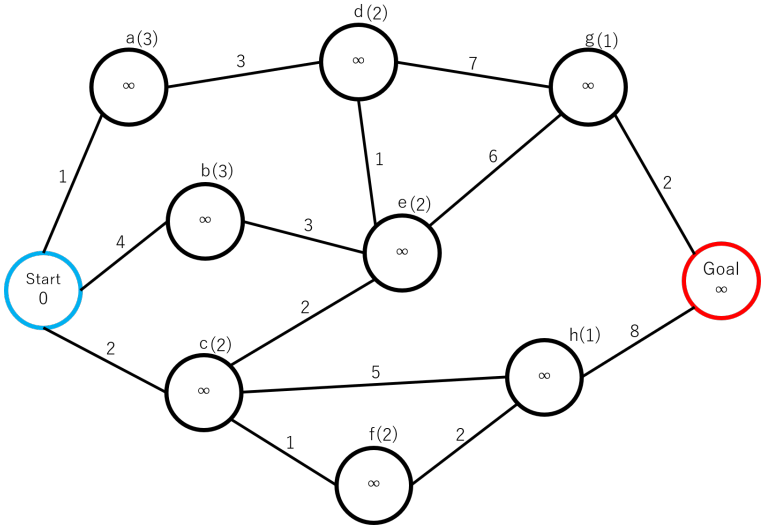

図 2.6　ヒューリスティックコストを追加したグラフ

2.2.3　D*アルゴリズム

D*アルゴリズムは、A*アルゴリズムを動的な変化に対応させたアルゴリズムである。このアルゴリズムでは、本来最短経路であったノード間の接続関係が無くなってしまった場合でも、関連するノードのみを更新することで効率的に処理を行うことが可能である。

2.2.4　コストマップの局所的な再設定による最短経路再探索の高速化手法

津川の手法は格子状のマップにおいて、D*アルゴリズムでは利点が発揮できない新たな経路が出現した場合でも効率的に処理を行うことが可能である。この手法の格子状のマップでは 1 マスごとをノードとして扱う。図 2.7 は格子状のマップを表したものである。ノードには、スタート地点となるスタートノード、ゴール地点となるゴールノード、通行可能の通常ノード、通行不可の壁ノードがある。図 2.7 では白いノードが通常ノード、黒いノードが壁ノードを表す。

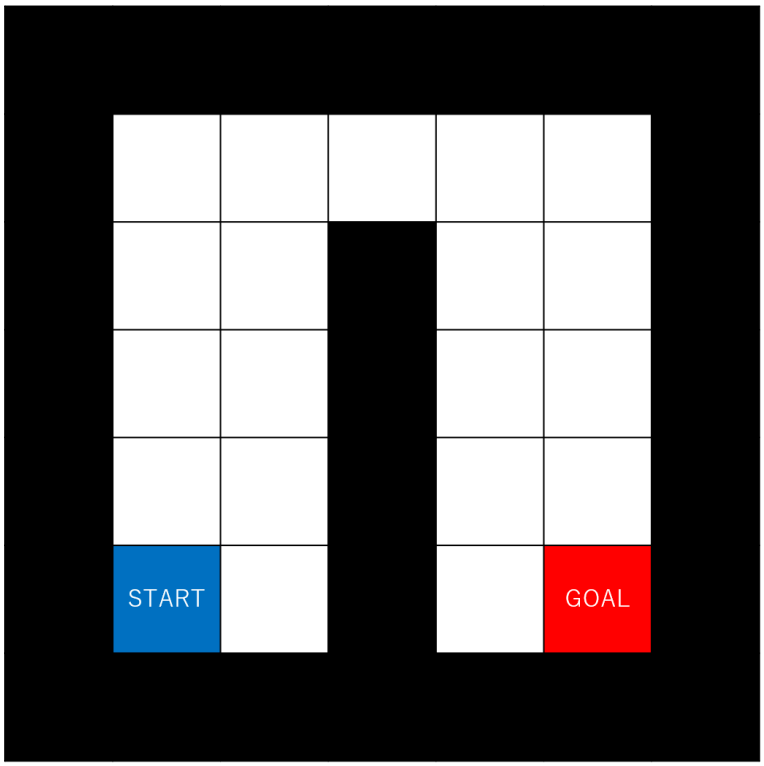

図 2.7　津川の手法における格子状のマップ

各ノードには、ゴールノードから各ノードまでのコスト、ゴールノードへの接続先であるノード ID、スタートノードから各ノードまでのコスト、スタートノードへの接続先であるノード ID の 4 つの情報が記録されている。ゴールノードへの接続先とは、スタートノードからノード a、ノード b、ゴールノードという経路を考えたとき、ノード a はゴールノードへ向かうため次にノード b を

通る。この場合はノード a にゴールノードへの接続先として、ノード b の ID を記録する。

各ノードに設定するコストはダイクストラ法でのコスト設定方法を用いる。またスタートノードへの接続先は、各ノードと隣接する上下左右のノードのスタートノードから各ノードまでのコストを比較し、コストの小さいノードのノード ID を記録しておく。ゴールノードへの接続先は、ゴールノードから各ノードまでのコストを同様の手順で比較することで設定する。この接続先であるノード ID はコストの大きいノードからコストの小さいノードへと向かう矢印として表現できる。図 2.8 はスタートノードから各ノードまでのコストと、スタートへの接続先を設定したコストマップを示す。これはスタートノードへの接続情報を示すコストマップである。また、図 2.9 はゴールノードからまでのコストと、ゴールへの接続先を設定したコストマップを示す。これはゴールノードへの接続情報を示すコストマップである。

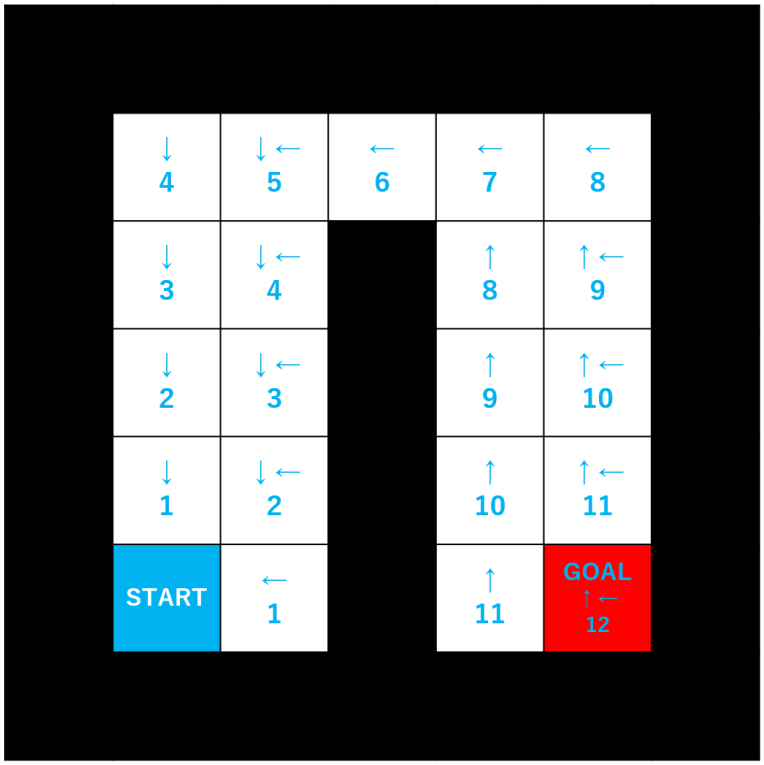

図 2.8　スタートノードへの接続情報を示すコストマップ

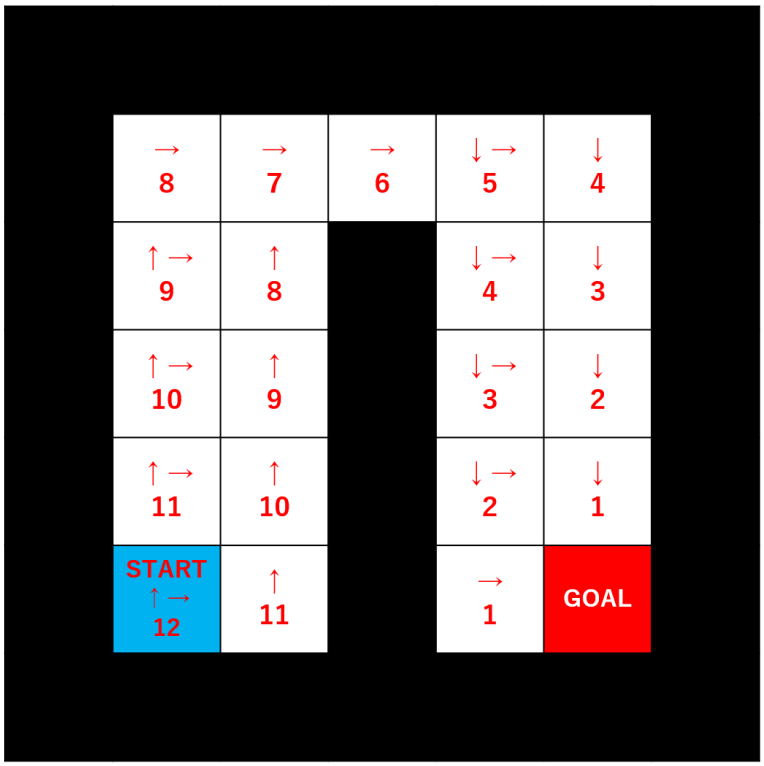

図 2.9　ゴールノードへの接続情報を示すコストマップ

次にマップが変化して新たな最短経路が現れた場合、コストマップを局所的に再構築する。図 2.10 は変化したマップを示す。

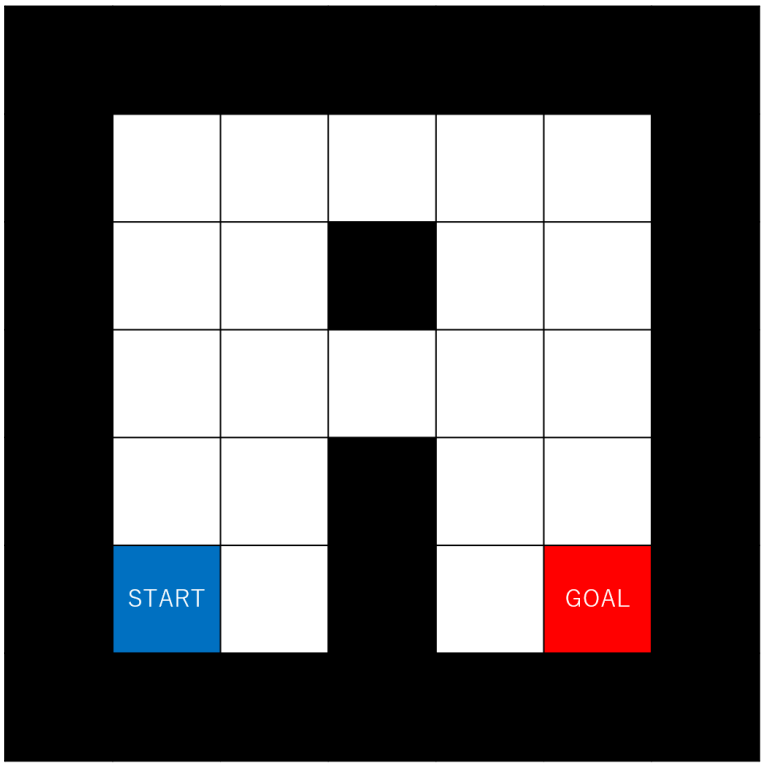

図 2.10　変化したマップ

まずスタートノードへの接続情報を示すコストマップにおいて、マップ変化により新たに通行可能となったノードと隣接する上下左右の通行可能なノードをみる。上下左右のノードの内、スタートノードにより近いノードを選択する。選択したノードのスタートノードまでの接続情報を元に、スタートノードから選択したノードまでの経路を記録する。図 2.11 は経路を記録した様子を示す。

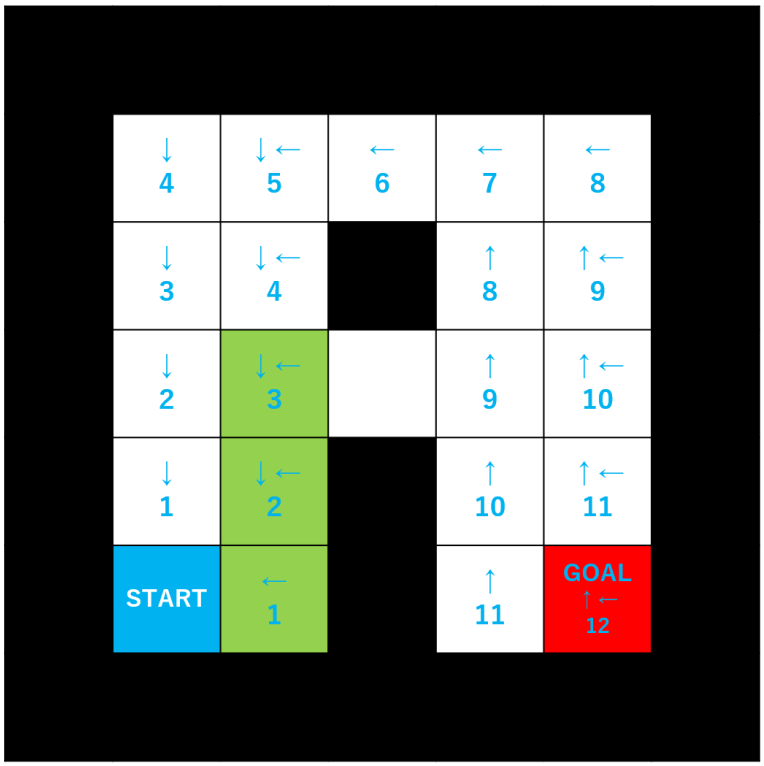

図 2.11　経路を記録した様子

次にゴールノードへの接続情報を示すコストマップにおいて、マップ変化により新たに通行可能となったノードと、先ほど記録した経路上のノードを更新する。図 2.12 はノードを更新した後の様子を示す。

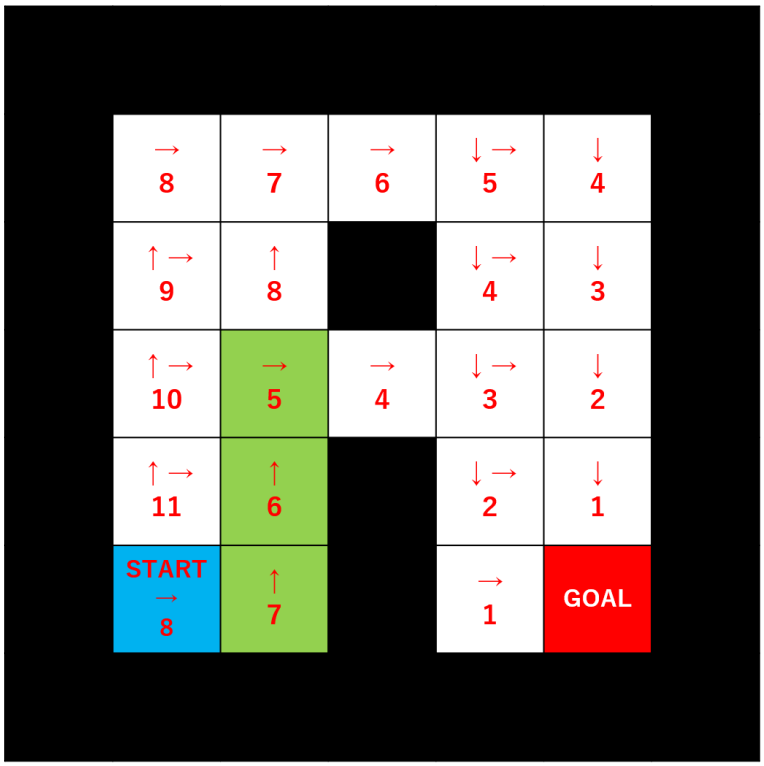

図 2.12　ノードを更新後のゴールノードへの接続情報を示すコストマップ

最後に、ゴールノードへの接続情報を示すコストマップにおいてスタートノードからゴールまでの接続情報を参照することで、新たな最短経路が求まる。図 2.13 は新たな最短経路が求まった様子を示す。

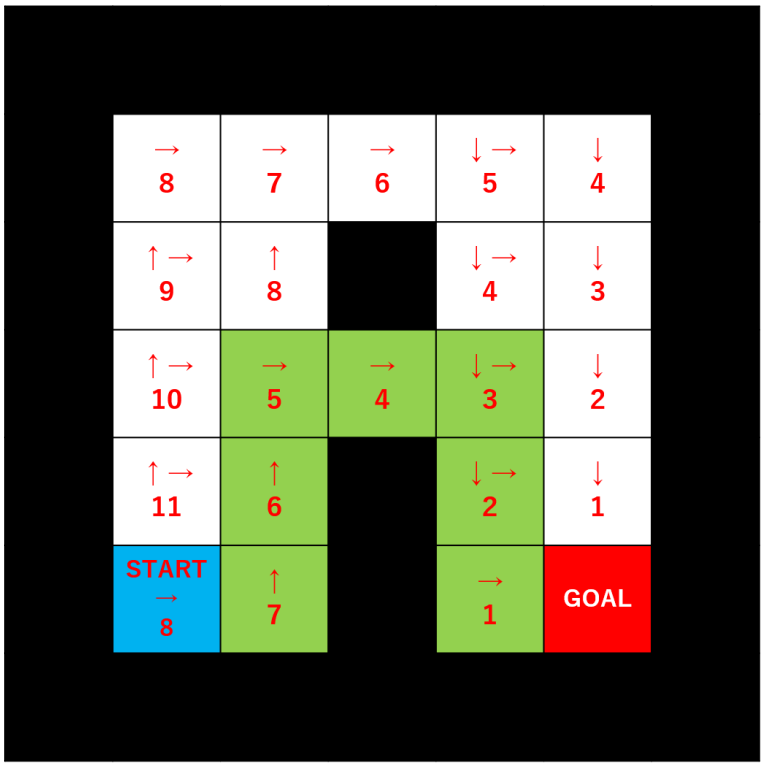

図 2.13　最短経路が決定した様子

コストマップを局所的に更新することで最短経路の算出を高速に行うことができる。しかし、コストマップ全体でみると整合性が取れない場合が出てくる。そのため、マップ変化時にコストマップ全体の更新も行う。コストマップ全体の更新と局所的な更新を並行して行うことで、高速化を図っている。

2.3　現状の問題点

2.2 節で述べた手法ではいくつかの問題点が挙げられる。

一般的にダイクストラ法や A*アルゴリズムは、マップ変化による経路変化が無い場合に効果的なアルゴリズムである。そのため、マップが動的に変化する場合、変化のたびに再探索を行う必要がある。

D*アルゴリズムは動的な変化に対応したアルゴリズムである。しかしマップ変化で経路が無く

なる場合には利点を発揮できるが、新たな最短経路ができた場合には利点が発揮できない。

津川の提案手法は、新たな最短経路ができた場合に対応したアルゴリズムである。しかし、この手法は、格子状のマップを前提にしており、ノードの接続数が任意である汎用的なマップには対応していない。

第 3 章

提案手法

本章では、提案する経路探索手法について説明する。3.1 節では初期コストマップの生成について説明し 3.2 節では、マップ変化によるコストマップの更新について説明する。3.3 節では、新たな最短経路の求め方について説明する。新たな経路が出現した場合に津川手法では対応できないグラフ理論における一般的なグラフにおいても対応出来る手法を提案する。

3.1　コストマップ全体の生成

本節では、グラフに対してコストマップを生成する手順について説明する。

まず、グラフのエッジに任意のコストを与え、各ノードに初期コストを設定することで、コストマップを生成する。本論文では、エッジのコストを 1 から 9 の整数をランダムで設定している。このコストマップの生成はダイクストラ法でのコストの設定方法を用いる。そして各ノードは、スタートからのコスト、スタートへの接続先情報、ゴールからのコスト、ゴールへの接続先情報、ノード ID の 5 つの情報を持つ。ノード ID とは各ノードを識別するためのものである。

あるノード α と接続関係にあるノード β において 2 つのノードのコストを比較する。接続関係にあるノード α のコストの方が値が大きい場合、ノード α はノード β のノード ID を接続先情報として記録する。本論文では、この接続先情報は接続関係にあるノードの内コストの小さいものを

記録していくので、コストの大きいノードからコストの小さいノードへ向かう矢印として表現する。図 3.1 は接続先情報を矢印で表したものである。図 3.1 ではノード a とノード d とそれらを繋ぐエッジがある。ノード a とノード d は接続関係にあり、それぞれのノードのコストは 1 と 4 である。2 つのノードのコストを比較するとノード d の方がコストの値が大きいため、ノード d はノード a のノード ID を接続先情報として記録する。また、接続先情報をノード d からノード a に向かう矢印として表現する。

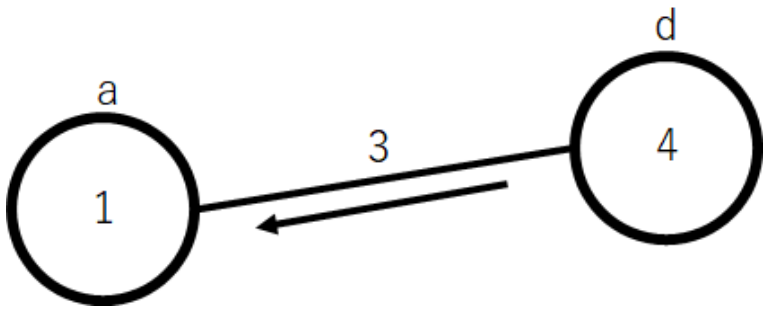

図 3.1　接続先情報を表した様子

スタートからのコストについて比較したものが、スタートへの接続先情報であり、ゴールからのコストについて比較したものが、ゴールへの接続先情報である。本論文では、スタートからのコストと、スタートへの接続先情報を示したコストマップをスタートマップと表記する。また、ゴールからのコストと、ゴールへの接続先情報を示したコストマップをゴールマップと表記する。図 3.2 はスタートマップを示し、図 3.3 はゴールマップを示す。スタートマップにおいてスタートノードとノード a は接続関係にある。スタートノードのコストは 0 であり、ノード a のコストは 1 である。この 2 つのノードのコストの値をみると、ノード a の方がコストが大きい。そのため、ノード a にスタートへの接続先として、コストが小さかったノードであるスタートノードのノード ID を記録する。すべての接続関係を持つノード同士に対して、同様の手順を行うことで、スタートからの接続先情報を記録する。ゴールからの接続先情報も同様の手順で記録する。

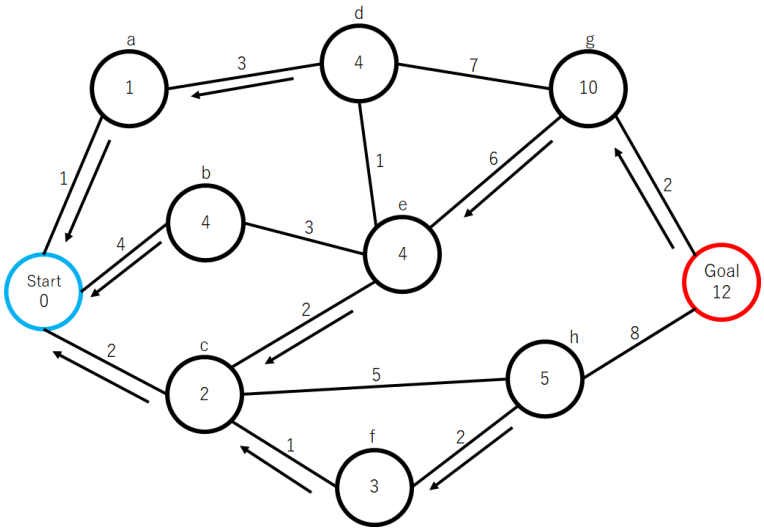

図 3.2　スタートマップ

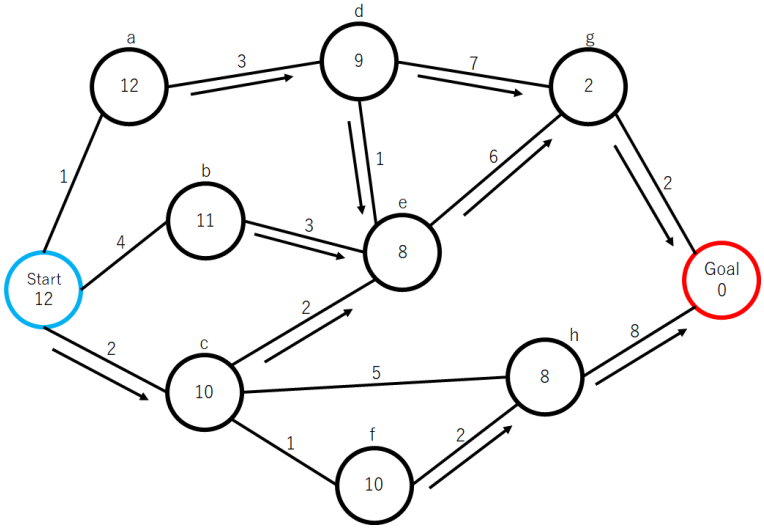

図 3.3　ゴールマップ

3.2　マップ変化によるコストマップの更新

本節では、マップが変化した際にコストマップの更新を行う手順について説明する。

ある 2 つの接続関係にないノード間に新たなエッジができた場合について考える。図 3.4 は図 2.1 に新たなエッジができたグラフを示す。

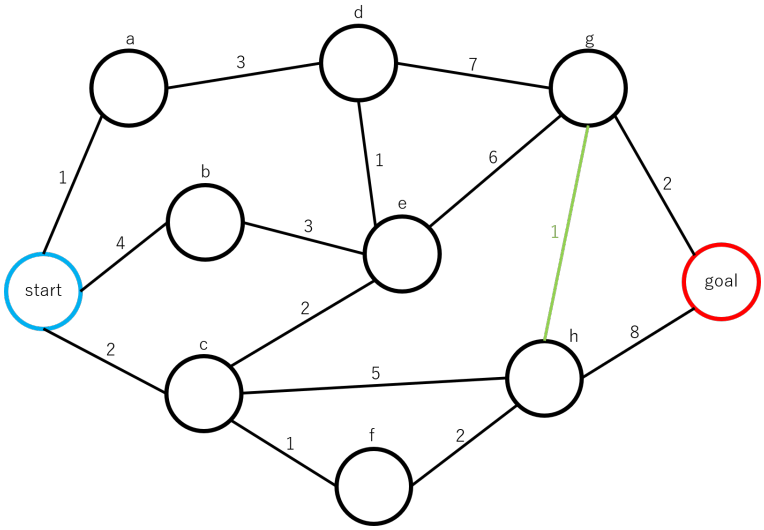

図 3.4　変化したマップ

マップが変化したことで新たな経路が出現した場合、局所的にコストマップを更新するために、更新するノードのノード ID を保持しておく配列データを用意する必要がある。スタートマップにおいてスタートへの接続先情報を保持する配列を A、ゴールマップにおいてゴールへの接続先情報を保持する配列を B とする。

スタートマップにおける新しいエッジと繋がれた 2 つのノードの内コストの小さい方を選択する。図 3.4 ではノード g とノード h が 2 つのノードに当たり、ノード h を選択する。次に選択したノードのスタートへの接続先情報をみる。選択したノードのコストと接続先情報のノードのコストの差が、2 つのノードを結ぶエッジのコストと同じ場合、配列 A にスタートノードへの接続先情報のノード ID を追加する。そして、最後に保持したノードのスタートへの接続先情報をみて同様の手順を繰り返す。上記の手順は配列 A にスタートノードのノード ID を追加するまで続ける。図 3.5 は配列 A の保持しているノード ID のノードを示す。図 3.5 では、スタートルートにノード h からノード f、ノード c、スタートノードのノード ID を保持している。

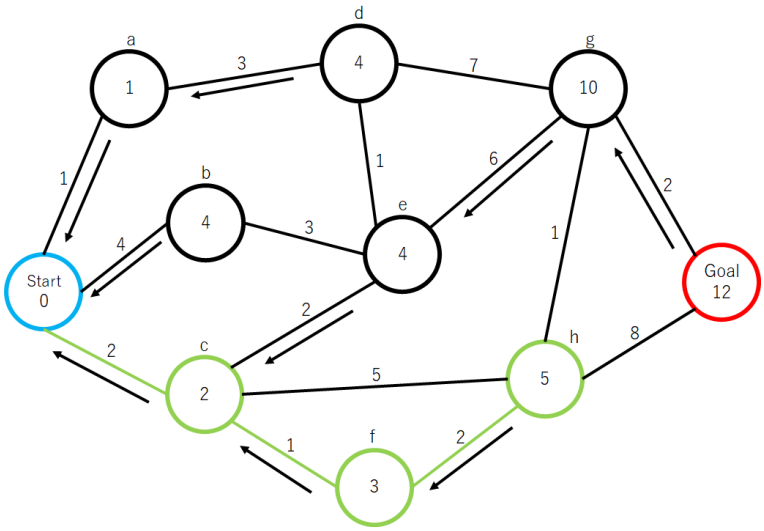

図 3.5　配列 A の保持しているノードを示した様子

次に、ゴールマップにおいて、配列 A が保持しているノードに対して更新判定を行う。この手順においてノードのコストが更新されなかった場合、新たに出現した経路は最短経路とならない。図 3.6 はゴールマップが更新された様子を示す。図 3.6 では、ノード h、ノード f、ノード c、スタートノードが更新されている。

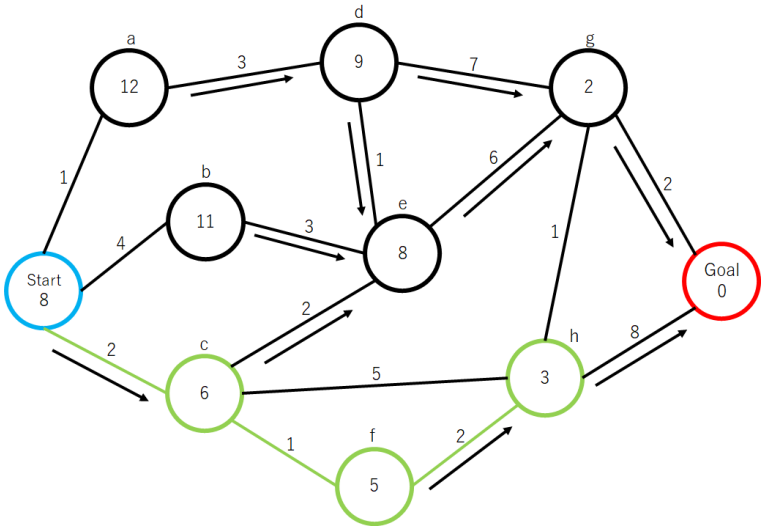

図 3.6　更新後のゴールマップ

3.3　新たな最短経路を求める

本節では、3.2 節で更新したゴールマップとスタートマップを元に、新たな最短経路を求める。

3.2 節で新たなエッジができた際、そのエッジに繋がれた 2 つのノードで選択しなかった方のノードをみる。そのノードのゴールマップにおける接続先情報をみる。そしてスタートマップで行った手順を同様に行い、配列 B にゴールノードのノード ID を追加するまで続ける。図 3.7 は配列 B の保持しているノード ID のノードを示す。図 3.7 では、ノード g とゴールノードを保持している。

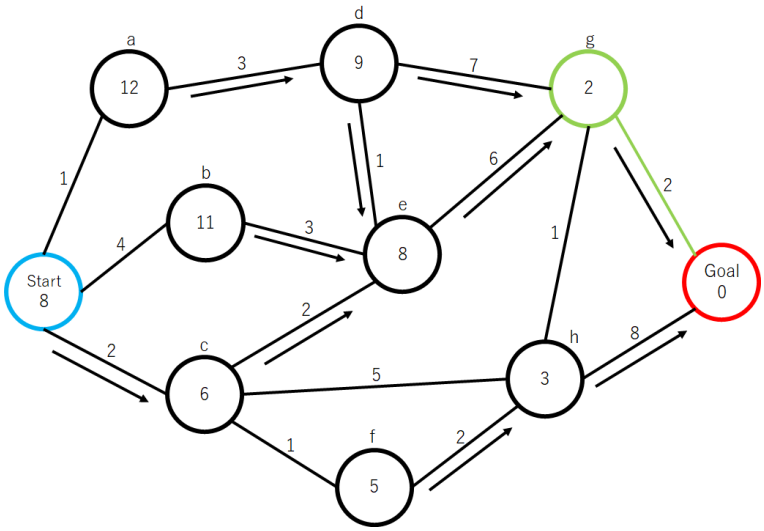

図 3.7　配列 B の保持しているノードを示した様子

最後に配列 A に追加したノード ID と、配列 B に追加したノード ID を元に新たな最短経路を求める。図 3.8 は求まった新たな最短経路を示す。図 3.8 では、スタートノードから、ノード c、ノード f、ノード h、ノード g、ゴールノードへと経路が求まる。

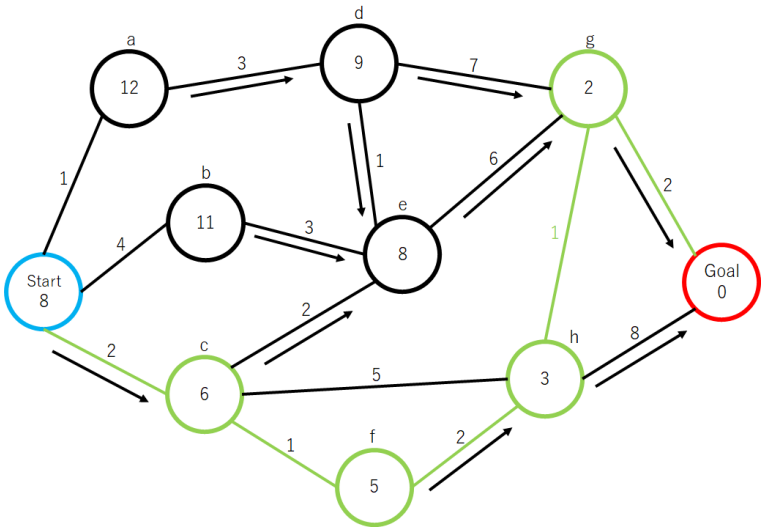

図 3.8　新たな最短経路

第 4 章

提案手法の検証

本章では、提案手法を用いた実験や実験結果、考察について述べる。4.1 節では、実験の方法について説明し、4.2 節では、実験結果について説明する。最後に、4.3 節では、考察を述べる。

4.1　既存手法との比較

提案手法を実装したプログラムについて説明する。図 4.1 と図 4.2 は提案手法を実装したプログラムを実行した様子である。図 4.1 ではエッジを追加する前の様子を示しており、図 4.2 ではエッジを追加した後の様子を示している。球状のオブジェクトがノードを表し、球同士を繋ぐ線がエッジを表す。青い球はスタートノード、赤い球はゴールノード、黒い球は通常ノードを表す。エッジのコストには 1 から 9 の整数をランダムに設定し、設定したエッジのコストを元にコストマップの生成、更新を行う。また、青色の線は新たなエッジが出現する前の最短経路を表しており、赤色の線は新たなエッジが出現した後での最短経路を表す。

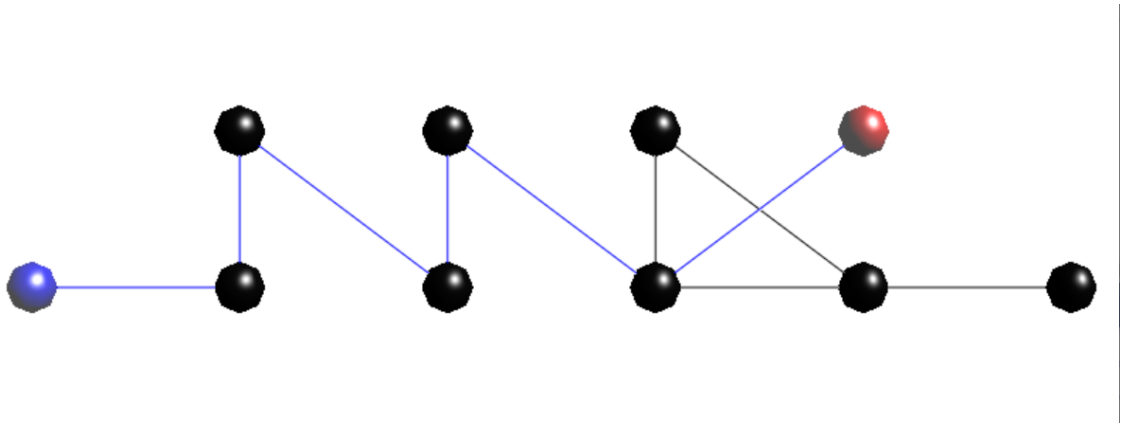

図 4.1　プログラムを実行した様子 エッジ追加前

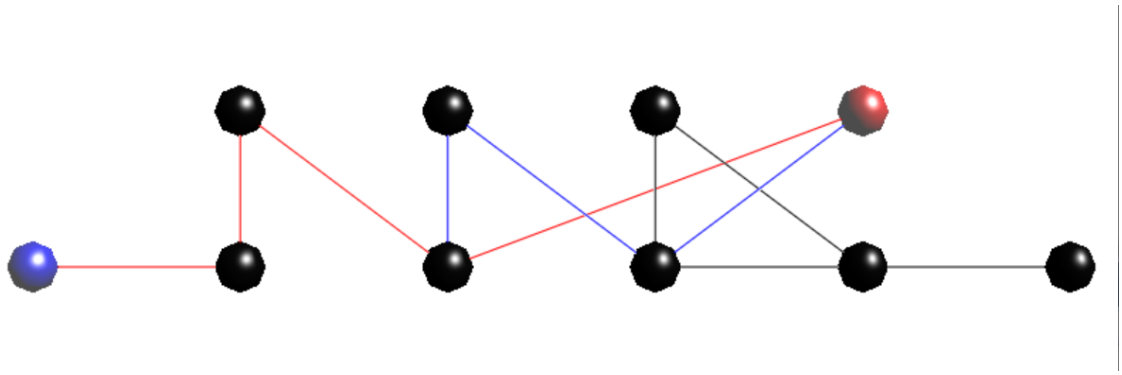

図 4.2　プログラムを実行した様子 エッジ追加後

マップ変化時の再探索を提案手法とダイクストラ法で行い、その処理時間を計測し比較する。提案手法については、マップ変化から局所的な更新を行い、新たな最短経路を求めるまでの時間と、マップ変化から局所的な更新、最短経路の算出を行い、コストマップ全体の更新を行うまでの時間の 2 つを計測する。

今回の実験で使用するマップは、ノード数の少ないマップとノード数の多いマップの 2 種類を用意した。ノード数の少ないマップはスタートノードとゴールノード含め、ノード数が 12 あるマップであり、図 4.3 はこのマップを示す。ノード数が多いマップはスタートノードとゴールノード含め、ノード数が 100 あるマップであり、図 4.4 はこのマップを示す。

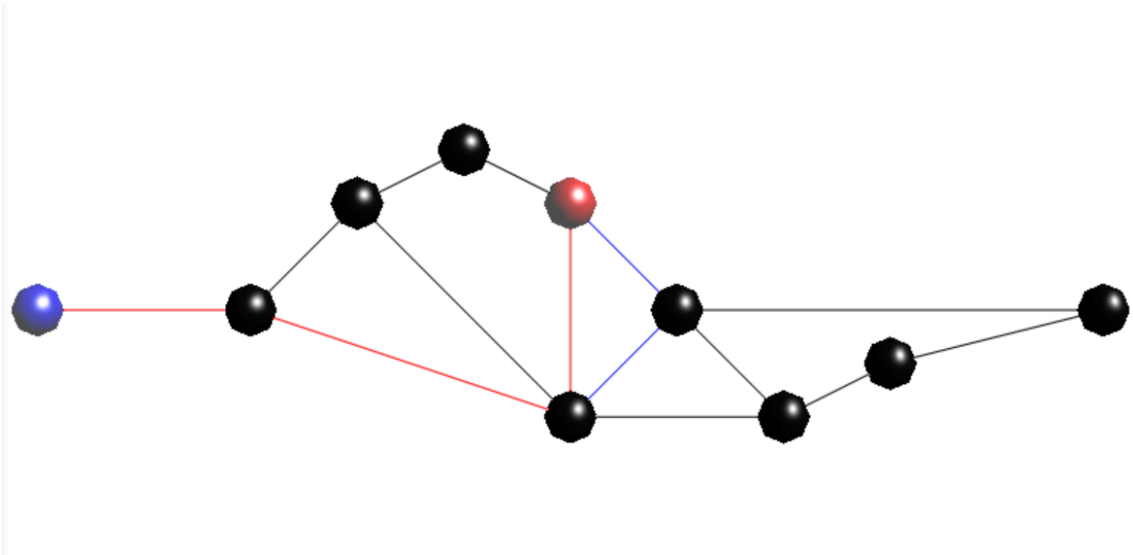

図 4.3　ノード数の少ないマップ

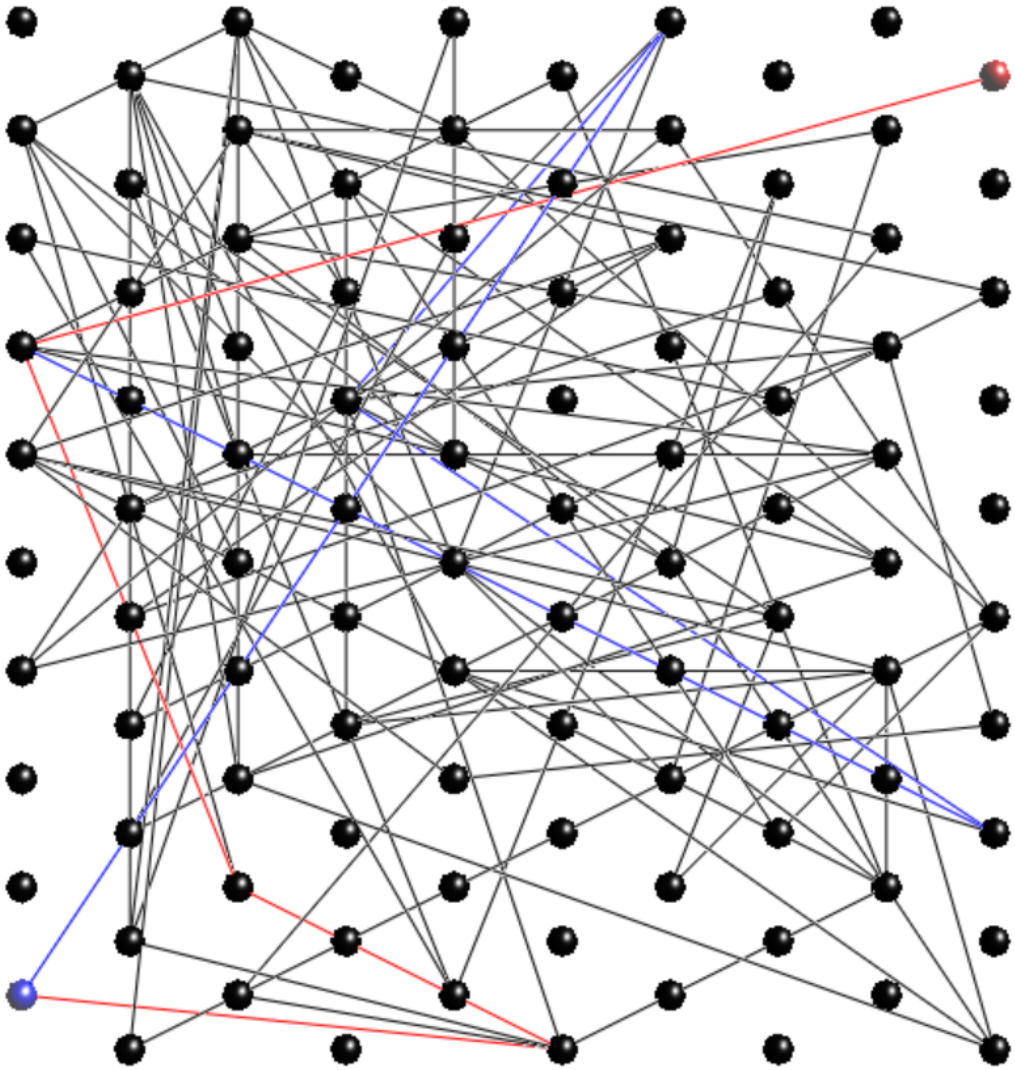

図 4.4　ノード数の多いマップ

4.2　実験結果

以下に、提案手法とダイクストラ法において、マップ変化時に再探索を行った際の処理時間を計測した結果を示す。表 4.1 はノード数の少ないマップでの結果を示し、表 4.2 はノード数の多いマップでの結果を示す。表 4.1、表 4.2 では、新たな最短経路を求めるまでの処理時間を提案手法 (局所)、コストマップ全体の更新を行うまでの処理時間を提案手法 (全体) と表記する。

表 4.1　ノード数の少ないマップ　実験結果

手法	処理時間 [ms]
提案手法 (局所)	0.034400
提案手法 (全体)	0.042100
ダイクストラ法	0.009000

表 4.2　ノード数の多いマップ　実験結果

手法	処理時間 [ms]
提案手法 (局所)	0.265700
提案手法 (全体)	0.343900
ダイクストラ法	0.316400

4.3　考察

提案手法は新たな最短経路を求めるまでの時間で見ると、ノード数の多いマップ、すなわち広いマップにおいてはダイクストラ法より処理時間が短いことが分かった。しかし、コストマップ全体の更新の時間を含めてしまうとノード数に関わらずダイクストラ法より処理時間が掛かってしまうことが分かった。津川の手法ではマップ変化後のコストマップの局所的な更新と、コストマップ全体の更新を並列的に処理することでこの問題の解決を図っている。ノード数が少ないマップにおいては、コストマップ全体の更新と局所的な更新で、更新を行うノードの数の差が少ない。また、提案手法ではスタートマップとゴールマップの 2 種類のコストマップを使用している。そのため、局所的な更新のみでもダイクストラ法より処理時間が掛かっている。

第 5 章

まとめ

本論文では、津川の提案した手法を元に、グラフ理論での一般的なグラフにおいて新たな経路が出現した場合に対応する経路探索手法を提案した。提案した手法は、津川の提案した手法のような格子状のマップだけでなく、各ノードの持つ接続関係が格子状のマップに比べて多い場合や少ない場合においても新たな最短経路を求めることができる。

また、マップ変化時に再探索を行った際の処理時間を既存手法であるダイクストラ法と比較した。ノード数の多い、すなわち広いマップにおける再探索ではコストマップを局所的に更新し、新たな最短経路を求める部分においてはダイクストラ法よりも短い処理時間であることが分かった。

本手法は、オープンワールドのような広いマップにおいて、マップ変化時の新たな最短経路の算出には有効である。しかし、その後の経路探索にはコストマップ全体の更新を行う為、津川の手法の様に並行処理で処理時間軽減を図る必要がある。

また、今回提案した手法では、マップ変化として新たなエッジを 1 つ追加した場合で検証を行った。そのため、新たなノードが出現した場合や、エッジを 2 つ以上追加した場合についての検証なども行う必要がある。

謝辞

本研究を進めるにあたってご指導いただいた先生方、先輩方や相談にのってくれた友人たちに感謝いたします。誠にありがとうございました。

参考文献

[1] 藤井叙人, 佐藤祐一, 中嶌洋輔, 若間弘典, 風井浩志, 片寄晴弘. 生物学的制約の導入による「人間らしい」振る舞いを伴うゲーム AI の自律的獲得. ゲームプログラミングワークショップ 2013 論文集, Vol. 2013, pp. 73–80, 2013.

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

[3] 柴田崇徳, 福田敏男, 小菅一弘, 新井史人. Genetic Algorithm を用いた移動ロボットの最適経路計画. 日本機械学会論文集 C 編, Vol. 58, No. 553, pp. 2714–2720, 1992.

[4] 独立行政法人工業所有権情報・研修館. 平成 16 年度　特許流通支援チャート　カーナビ経路探索技術. http://www.inpit.go.jp/blob/katsuyo/pdf/chart/fdenki22.pdf. 参照:2019.11.25.

[5] Linkedin Corporation. NAVITIME の経路はこうして作られる. https://www.slideshare.net/NavitimeJapan/navitime-89761183. 参照: 2020.01.19.

[6] 北原武嗣, 岸祐介, 久保幸奨. 高低差を考慮した津波災害時の群衆避難における経路選択に関する一検討. 土木学会論文集 A1(構造・地震工学), Vol. 69, No. 4, pp. 1067–1075, 2013.

[7] 姫野湧太, 徳永潤平, 榎原博之, 上田修功. ベイズ最適化を用いたフロー型実時間避難計画.

情報処理学会　研究報告数理モデル化と問題解決 (MPS), Vol. 2019-MPS-125, No. 10, pp. 1–6, 2019.

[8] 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.

[9] Unity Technologies. ナビゲーションと経路探索. https://docs.unity3d.com/ja/current/Manual/Navigation.html. 参照: 2020.01.15.

[10] KADOKAWA Game Linkage. いかにしてサーバーはモンスターを歩かせるのか？ 「ファイナルファンタジー XIV:新生エオルゼア」の経路探索テクニック【SQEX オープンカンファレンス 2012】. https://www.famitsu.com/news/201211/29025006.html. 参照: 2020.01.15.

[11] E.W.Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, Vol. 1, pp. 269–271, 1959.

[12] Jorudan. 乗り換え案内　ジョルダン. https://www.jorudan.co.jp/norikae/. 参照: 2020.01.14.

[13] 森畑明昌, 松崎公紀, 武市正人. 乗換え案内サービスにおける経路探索手法. 電子情報通信学会論文誌 D, Vol. J88-D1, No. 10, pp. 1525–1533, 2005.

[14] Peter E. Hart, Nils J. Nilsson, and Bertram raphael. A formal basi for the heuristic determination of minimal cost paths. *IEEE transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107, 1968.

[15] Howie Choset. Robotic motion planning a* and d* search. https://cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf. 参照:2019.11.25.

[16] 津川巧, 阿部雅樹, 渡辺大地. コストマップの局所的な再設定による最短経路再探索の高速化手法についての研究. 芸術科学会 NICOGRAPH2019, 2019.

[17] 松田喬, 元田浩, 鷲尾隆. 一般グラフ構造データに対する Graph-Based Induction とその応用. 人工知能学会論文誌, Vol. 16, No. 4, pp. 363–374, 2001.

[18] 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.

[19] ltd astamuse company. コストマップの意味・用法を知る. https://astamuse.com/ja/keyword/10176006. 参照: 2019.11.25.

[20] 平石広典, 大和田勇人, 溝口文雄. 実用的な経路計画生成のための時間制約付きヒューリスティック探索. 情報処理学会論文誌, Vol. 40, No. 11, pp. 4021–4029, 1999.