評価関数の更新を伴う探索手法に関する研究

平成 26 年 9 月

和歌山大学大学院システム工学研究科
芦田昌也

Studies on Informed Search Methods with Improving Evaluation Functions

by
Masaya Ashida

Graduate School of Systems Engineering
Wakayama University

September, 2014

概要

本論文は，探索品質の向上を目的として，木構造で表される状態空間を対象に，評価関数の更新を伴う発見的探索の手法について論じたものである．

状態空間に付されたコストを用いて，探索により発見される解の品質を評価する．また，解を発見するまでの展開状態数や生成状態数を用いて，探索過程の品質を評価する．これらで測られる探索品質には，コスト関数とヒューリスティック関数で構成する評価関数の性質が関与している．探索品質の向上には，コストの特徴を反映した発見的知識を利用し，適格性を有した精度の高い評価関数とすることが効果的である．

評価関数の精度については，次の二つの観点がある．ひとつは，コストの真値と評価関数による推定値との誤差に着目するものである．もうひとつは，コストの真値の大きさで決まる解の順序関係を反映する能力に着目するものである．ヒューリスティック関数が注目状態から隣接状態までのコストを推定する短期予測と，隣接状態から目標状態までの長期予測の 2 項で構成されるとき，これら 2 種類の予測精度を同程度の水準に保つことが，最適解と非最適解の正確な識別につながる．

コストが経路に沿って一様に増加する状態空間においては，このコストの特徴を反映した発見的知識に基づいて，ヒューリスティック関数を更新する．その更新方法は，ひとつの状態の部分木内を対象として，適格性を損なうことなく，コストの真値と推定値との誤差を減少させるものである．この更新方法と A*アルゴリズムとを組み合わせた探索手法は，A*アルゴリズムと同様に最適解の発見を保証する．また，生成状態数に由来する有効分岐因子の値と展開状態数とは，ともに A*アルゴリズムのそれ以下である．

展開済みの状態から新たな隣接状態が生成される場合や，確定値に到達するまでコストが変化する状態空間では，状態の初回の展開時にコストの真値が確定しないことがある．そのような状態空間では，同一状態の再展開を許容し，新たに獲得した情報を利用してコスト関数を更新する．同一状態の再展開とコスト関数の更新を伴う探索手法は，A*アルゴリズムの適格性に類似する条件を満たす場合に最適性を有する．また，経験的に得られる情報をヒューリスティック関数に反映させることで，展開状態数の減少が見られる．

これらは，状態空間の特徴を考慮しつつ評価関数を適切に更新することが，探索品質の向上に寄与することを示すものである．

Abstract

This thesis discusses informed search methods with improving evaluation functions for a state space described as weighted tree structure. The purpose of this research is enhancement of quality for heuristic search.

The search quality is measured by cost of the solution, the number of expanded states and the number of generated states. It is effective for the enhancement of search quality that the accuracy of evaluation functions is improved by heuristics to maintain admissibility.

There are two orientations to improve the accuracy of evaluation functions composed of cost function and heuristic function. One is to reduce the error between actual cost and the estimated cost by evaluation functions. Another is to reflect the order of solution cost defined by the actual cost. In dividing a heuristic function into two subfunctions, it provides accurate distinction between the optimal solution and non-optimal solutions to keep the same level of the accuracy of these two subfunctions.

On a state space that cost of each edge along a path increases uniformly the heuristic function is updated based on the heuristics related to the cost characteristics. The update method changes cost in the subtree for a state and reduces the error between actual cost and estimated cost with maintaining admissibilty. A proposed search method combined the update method with A* search ensures the finding the optimal solution. In the proposed method both the number of expanded states and the value of effective branching factor are less than or equal to those in A* search.

The actual cost is not always determinate in visiting a state at first time on a state space that a new state appears from a state already expanded and that cost changes until the definite value. On that kind of state space a proposed search method similar to A* search with revisiting states and updating the cost function using newly acquired infomation through the revisiting. The proposed method has optimality under the similar condition to admisibilty.

Improvement of evaluation functions under consideration of the cost characteristics on the state space is effective to enhancement of search quality.

目次

第 1 章

序論

1.1 背景

探索は問題解決を行うための枠組みのひとつである．そこでは，解決すべき問題が状態空間として表現される [1]．状態空間は，その問題に内在する様々な状態の集合であり，状態間の可能な遷移関係を表現したものである．問題解決を開始する直前の状態に対応する初期状態から，問題が解決された状態に対応する目標状態に至るまでに必要な操作や状態遷移の系列に対応した経路や，発見された目標状態そのものが解である．探索は，そのような解を系統的に発見しようとする手法である．解を発見する上で，問題解決の途中の状態において，解決の過程（状態遷移過程）を試行錯誤で探すことがある．また，それらの解決の過程が，時間，労力，費用などのコストを要する場合は，できる限りコストの小さい解が望まれる．解が発見されることにより，問題が解決されたものとみなされる．

探索の手法は，情報を利用しない探索手法と情報を利用する探索手法に大別される．ここで利用される情報とは，解決すべき問題の定義には含まれない，その問題に固有の発見的知識である [2].

情報を利用しない探索手法において，コストを考慮しない場合は，状態空間を網羅的に探査する探索戦略の下で解の発見が試みられる [3] [4]．有限の状態空間においては，すべての状態を探査できるため，解が存在すればその発見を保証できる．しかし，探索戦略によっては，解の発見までに多くの時間や記憶域を必要とする場合がある．

コストを考慮する場合は，通常，状態空間においてコストの小さい経路から選択的に探査しながら，最適解としてコストが最小となる解の発見を試みる．これにより，結果として探索途中で最適解より大きなコストを有した経路を探査することがなくなる，このよう

な選択的な探索戦略は，解を発見するまでの時間や記憶域を削減する可能性がある．

情報を利用する探索手法では，コストを考慮する問題が対象となる．情報を利用してコストを推定し，そのコストの小さい経路から選択的に探査しながら解の発見を試みる．そのような手法の中に，コストの推定値が正確なコストを超えない，適格性と呼ばれるある一定の条件が満たされている場合には，最適解の発見を保証する探索手法が存在する．そこでは，コストの推定値の精度が高まるにつれ，解の発見に必要な時間や記憶域の削減可能性が示唆されている [5]．

このことから，情報を利用する探索手法の研究は，問題解決の品質を高めるために，問題解決に有効な問題固有の発見的知識を導入すること，発見的知識を推定値に反映させる上で適格性を満たすこと，さらに，より精度の高い推定値を付与することに注力している [6] [7]．

探索の典型的な例題には，古くからパズルが用いられる．その多くは，状態空間の構造や正確なコストが本質的には既知である．しかし，問題解決にあたって，状態空間の構造や正確なコストが付与されるとは限らない．そのため，状態空間において探査を終えた部分に関しては正確なコストを獲得し，未探査の部分については不確かさを残して探索が進められる．探索の途中で状態空間の構造やコストに変化が生じることはないため，発見した解は問題解決を行う時点においても有効である．

探索の際に情報収集のための観測や行動を伴う問題では，状態空間の構造や正確なコストが必ずしも明確にはならない場合がある．そこでは，状態空間において一旦は探査を終えた部分に未探査の状態が追加されることや，一回の探査で正確なコストを獲得できないことなどが生じる．そのため，状態空間全体にわたり不確かさが残るため，再探査も行いつつ探索が進められる．発見した解は，探索終了後に状態空間の構造やコストに生じた変化のために，問題解決を行う時点には必ずしも有効ではないことがある．

このように，探索手法の研究においては，問題解決の対象となる問題領域を拡大する上で，状態空間の性質に適合する探索手法を提案するとともに，その手法による探索の品質を明確にすることが行われてきた．本研究は，コストの推定値の性質と精度が探索の品質に関与することを踏まえ，コストの推定値を与える評価関数の精度向上のための方策，状態空間の性質を考慮した探索手法の提案，および，提案手法による探索の品質という点から，評価関数の更新を伴う探索手法について議論するものである．

1.2 研究の目的と課題

本研究の目的は，不確かさのある状態空間における探索の品質向上を目指すことにある．対象とする状態空間はコストが付された木構造である．ここでの不確かさとは，状態空間に付されるコストの正確な値が事前にはわからないことを意味するものである．探索の品質のひとつは，発見される解に関する最適性の有無である．コストに設けた基準に照らし解の優劣を評価する場合に，もっとも優れた解の発見を保証できるかどうかである．もうひとつは，解を発見する過程に関する品質であり．探査する範囲の広さにより評価されるものである．すなわち，探索の品質としては，最適解の発見を保証することに加え，可能な限り無駄な探査を行わないことが求められる．

最適解の発見を保証するような性質を有した評価関数であっても，推定値の精度によっては，最適解に至る経路上にない状態を数多く探査する可能性がある．一方で，まったく無駄なく探査を進め解に至るような性質を有した評価関数であっても，発見した解が最適解ではない場合もある．このため，探索の品質を向上させるためには，最適解の発見を保証する性質を維持しつつ，精度の高い評価関数を構成することが重要であり，そのための指針を示すことが課題となる．

本研究では，コスト関数とヒューリスティック関数とにより評価関数を構成する．コスト関数は，状態空間の探査済みの範囲においてコストを推定するものであり，ヒューリスティック関数は，未探査の範囲のコストを推定するものである．評価関数の値に基づいて選択的に探査を行う上では，評価関数がコストの真値による解の評価の序列を反映する性質を有していれば，評価関数値とコストの真値との間に誤差が生じていても最適解の発見が可能である．より弱い条件としては，最適解以外の解の序列を正確に反映する必要はなく，最適解がそれ以外のすべての解より上位にあればよい．本研究においてこの観点から検討し，ヒューリスティック関数を 2 つの副関数に分割し，それらの精度を同程度に保ちながら探索することにより，最適経路からの逸脱を抑制できる可能性があることを示唆する．

評価関数の精度を高める直接的な方法は，評価関数値とコストの真値との誤差を減少させることである．不確かさのある状態空間においては，探索の過程で事前に得られなかった情報を獲得できることがある．そのような情報を利用して評価関数を更新することにより，誤差を減少できる可能性がある．この観点から，本研究では状態空間の性質を考慮

し，評価関数を更新しながら探索を行う手法を提案する．その手法において，最適解の発見が保証されることと，探索中に現れる状態数の減少が見込まれることを示す．

提案手法のひとつは，コストの特徴が既知である状態空間において，その特徴を活用するための発見的知識と獲得できるコストの真値を利用してヒューリスティック関数を更新しながら探索を行う手法である．最適解の発見が保証されるとともに，代表的な手法である A*アルゴリズムを対照手法とした場合に探索中に現れる状態数の減少が見込まれる．

もうひとつの提案手法は，観測データに対して生じる多義的な解釈を状態空間として，事前に付与される情報との整合性を評価しながら探索を行う手法である．新たな観測データが追加され，状態空間の構造が変化することに対応してコスト関数を更新する．最適解の発見が保証されるとともに，実験的に得られるコストの収束値を用いてヒューリスティック関数の精度を調整することにより，探索中に現れる状態数の減少が見込まれることを示す．

これらの結果を踏まえ，状態空間の性質を考慮しつつ評価関数を更新することが探索の品質向上に寄与するものであることを述べる．

1.3　本論文の構成

第 2 章では，探索手法を用いた問題解決について述べる．代表的な探索手法を概観し，探索戦略により探索手法を分類する．また，コストが付された状態空間を対象として，探索中に得られるコストの評価値の性質により状態空間を分類する．これらの分類の下で，探索効率を向上させるためのアプローチを述べる．

第 3 章では，評価の高い経路から優先的に探索する探索手法によって発見される解の性質に対する評価関数の精度の影響について述べる．ヒューリスティック関数において，注目頂点に隣接する頂点までのコストの評価値の精度と，それら隣接頂点から目標頂点までのコストの評価値の精度との関係から，発見される解の性質について議論する．

第 4 章では，コストの特徴が既知である状態空間での探索手法について述べる．局所的に獲得した実コストと，コストの特徴を活用するための発見的知識とを利用して，ヒューリスティック関数を更新することにより，同一状態空間において，A*アルゴリズムを使用する場合より有効分岐因子が小さくなることが示される．

第 5 章では，探索による問題解決のひとつのアプリケーションを示す．情報が不足することにより生じる多義的な解釈の関連を状態空間として表現し，情報収集のための行動計

画の立案と，整合性のある解釈の獲得のために探索が行われる．提案手法により，最適解として整合性のある解釈を発見できることが示される．

第 6 章では，本研究の結論と今後の課題を述べる．

第 2 章

探索を用いた問題解決

2.1　状態空間の探索

探索は，解決の対象とする問題を表現した状態空間を利用して，その初期状態から目標状態に至る過程を探すことで，問題解決をはかる基礎的な技術である．問題解決の過程において生じる様々な状況を離散化して表現した**状態**と，問題解決のためにある状態から別の状態へ遷移させる**作用素**とにより，問題を表現したものが**状態空間**である．

図 2.1 に 8 パズルの状態の例を示す．8 パズルは，1 から 8 までの数が書かれた駒を 3×3 のます目のある盤面に配置し（図 2.1（a)），空いたます目に駒をスライドさせて配置を変更しながら，数の順に並べ替えるパズルである（図 2.1（b)），

このパズルの状態空間における状態は，駒が配置された盤面である．駒をスライドさせると盤面が変化し，異なる状態に遷移する．駒の移動が空いたます目の移動とみなせることに着目し，空いたます目の移動を作用素とする．図 2.2 に 8 パズルの場合の状態空間の一部分を示す．right，up，down，left は空いたます目の移動方向である．

状態空間を利用した問題解決とは，現在の状況に対応する**初期状態**から，問題が解決された状況に対応する**目標状態**へ遷移するための作用素の系列を発見することである．その

1	5	2
7	4	3
8	6	

1	2	3
4	5	6
7	8	

（a）初期状態の例　　（b）目標状態の例

図 2.1　8 パズルの初期状態と目標状態の例

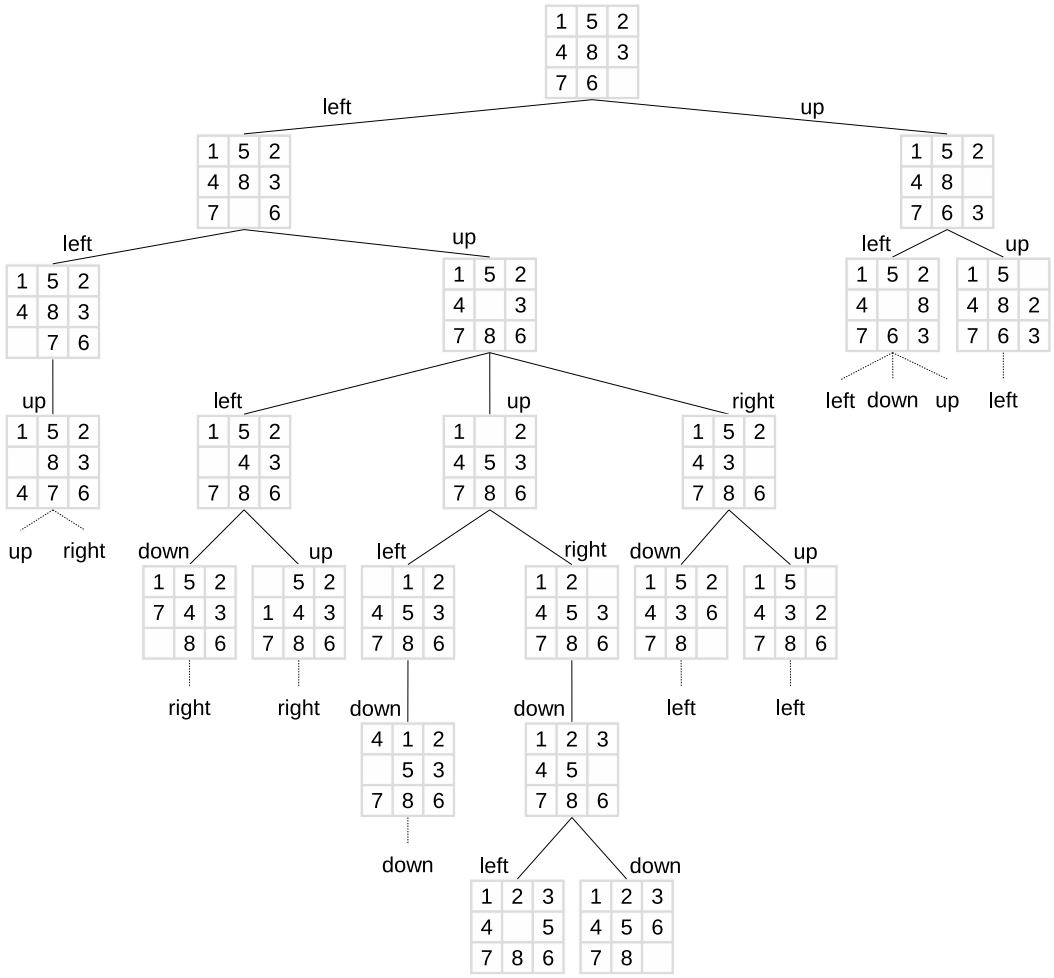


図 2.2　8 パズルの状態空間の例の一部分

ために，作用素の適用によって生じる状態遷移を試行しながら，目標状態を発見しようとする過程が**探索**である．その手法は，適用される状態空間の性質，発見できる解の性質，問題固有の情報利用の有無，求解の過程での探索の振る舞いなどにより特徴づけられる．

状態空間の数学的記述には，**グラフ**が利用される（図 2.3）．状態はグラフの**頂点** v で表現され，作用素はグラフの**辺** (v_i, v_{i+1}) に対応付けられる．グラフ構造上での探索は，初期状態に対応する初期頂点 v_0 から，辺で接続された頂点 v_i を探査しながら，目標状態に対応する目標頂点 v_n に至る**経路** $(v_0, v_1, \cdots, v_n)$ を発見する手法に帰着される．発見された経路が探索の**解**であり，その経路の初期頂点から目標頂点に至る構成辺に対応する作用素の系列が，問題解決のための手順に対応する．

問題解決の目的によっては，作用素の適用コストを考慮することがある．作用素の適用コストを状態遷移の**コスト**とみなし，グラフ上では辺の重み $w(v_i, v_{i+1})$ として表現する．

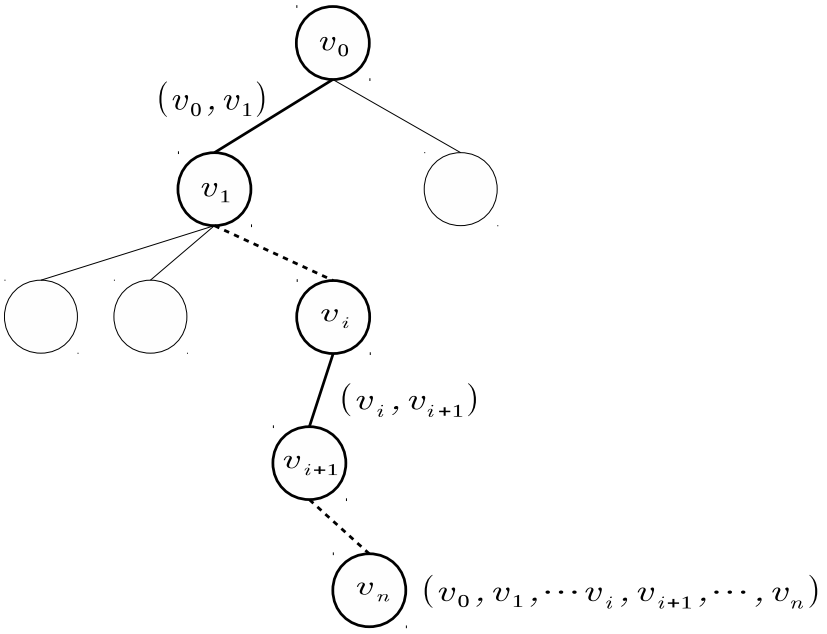


図 2.3 状態空間の模式図と表記

経路の構成辺の重みの和をその経路のコストとすることで，解の優劣をコストの値により評価できる．一般にはもっともコストの小さい解が最適解である．

探索の基本的な枠組みは，次の一連の処理の繰り返しである．まず，探査対象となる頂点リストの先頭にある頂点を**注目頂点**として取り出し探査する．もし，注目頂点が目標頂点であれば，解を発見したものとして探索は終了する．そうでなければ，注目頂点を展開し辺で接続された隣接頂点を生成する．展開した注目頂点は，探査済みの頂点リストに格納する．次に，生成された隣接頂点のそれぞれについて，一定の条件を満たしている頂点のみ，探査対象となる頂点のリストに格納する．探索戦略にあわせて探査対象となる頂点リストに格納されている頂点を整列する．こののち，再び注目頂点の選択に戻ることを繰り返し，解を発見して終了するか，もしくは，探査対象となる頂点がなくなれば終了する．

図 2.4 にこの枠組みを示す．頂点と頂点リストはいずれもオブジェクトとして表記する．頂点オブジェクトに含まれるメソッドとプロパティを表 2.1 に示す．頂点リストオブジェクトに含まれるメソッドを表 2.2 に示す．

探査対象となる頂点リストと探査済みの頂点リストは，一般にそれぞれ OPEN **リスト**，CLOSED **リスト**と呼ばれている．初期頂点を v_0，注目頂点を v，v を展開して生成される頂点集合を $\{u_1, \cdots, u_n\}$，OPEN リストを $OPEN$，CLOSED リストを $CLOSED$ とする．

表 2.1　頂点オブジェクト

メソッド	
generate()	当該頂点の隣接頂点を生成する
isGoal()	当該頂点が目標頂点であれば True を返し，目標頂点でなければ False を返す
isExplore()	当該頂点が探査対象になる場合は True を返し，探査対象にならない場合は False を返す
プロパティ	
length	初期頂点からの経路長を格納する

表 2.2　頂点リストオブジェクト

メソッド	
pop()	当該リストの先頭要素を返す
add(v)	頂点オブジェクト v を当該リストに追加する
sort()	当該リストに格納している頂点オブジェクトを探索戦略に合わせて整列する
isMember(v)	当該リストが頂点オブジェクト v を含む場合は True を返し，v を含まない場合は False を返す
isNotEmpty()	当該リストが空でない場合は True を返し，空である場合は False を返す

問題解決に必要な情報がすべて与えられる問題は，状態空間に含まれるすべての状態と作用素，および作用素の適用コストが既知である．このような問題は，不確かさのない状態空間で表現される．これに対し，事前には明示されない状態が存在する場合や，作用素のコストが探索中には正確にはわからない場合など問題解決に必要な情報が不足する問題は，不確かさのある状態空間として表現される．問題解決に必要な情報の与えられ方により状態空間の性質が変わるため，それに対応した探索の方法が求められる．

```
1  OPEN.add(v_0);
2  while OPEN.isNotEmpty() do
3  |  v ← OPEN.pop();
4  |  if v.isGoal() then
5  |  |  return(v);
6  |  end
7  |  {u_1, ··· , u_n} ← v.generate();
8  |  CLOSED.add(v);
9  |  for u ∈ {u_1, ··· , u_n} do
10 |  |  if u.isExplore() then
11 |  |  |  OPEN.add(u);
12 |  |  end
13 |  end
14 |  OPEN.sort();
15 end
16 return(False);
```

図 2.4 探索の基本的な枠組み

2.2 不確かさのない状態空間における探索手法

コストを考慮しない不確かさのない状態空間では，すべての頂点を網羅的に探査することにより解を発見できる．その基本的な手法のひとつである**深さ優先探索**では，OPEN リストで探査対象の頂点を後入れ先出し法 (LIFO:Last In First Out) で管理し，全ての状態をもれなく探査する（Nilsson [8] によると，この手法は Solomon [4] に記されている）．

同様に，網羅的に探査する手法である**幅優先探索**は，OPEN リストで探査対象の頂点を先入れ先出し法 (FIFO:First In First Out) で管理するものである．迷路を抜ける経路を求める手法として Moore [3] により定式化されている．

解を発見するまでに，深さ優先探索は時間計算量が多く，幅優先探索は空間計算量が多いことが指摘されている．これらの弱点は，OPEN リストに追加する頂点について初期

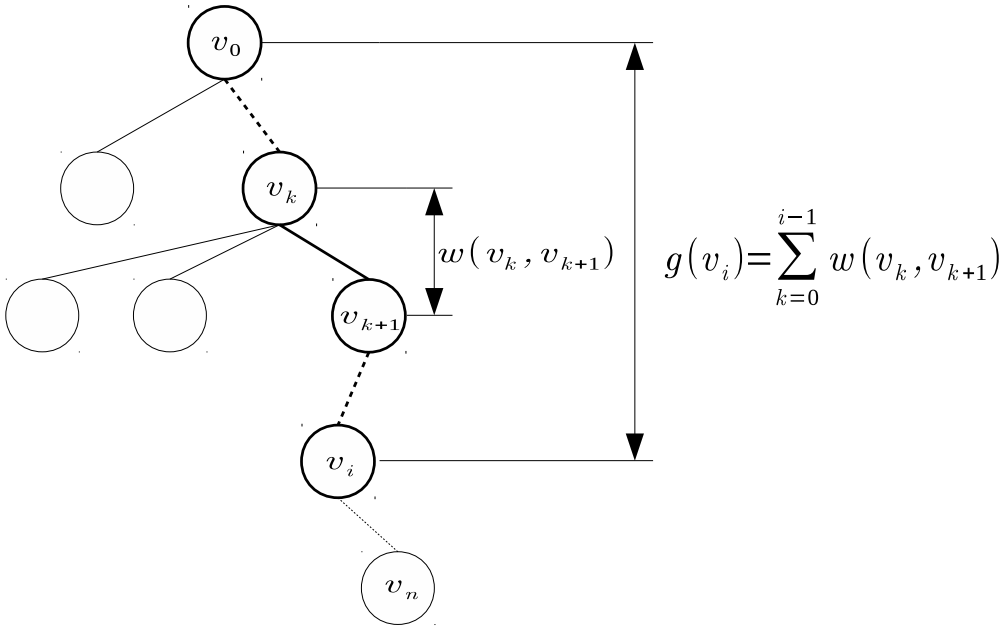


図 2.5　コスト関数

頂点からの経路長による制限を設け，解が見つからない間は経路長の制限を緩和しながら深さ優先探索を繰り返し実行する**反復深化深さ制限探索**によって回避されることが示されている [9].

作用素のコストとして問題に応じた特徴量を対応付けることにより，対象とする問題領域を拡張することができる．コストを考慮する状態空間では，単に初期状態から目標状態に到達する経路を解として発見するだけでなく，発見した解の優劣をコストを利用して評価できる．評価関数のひとつとして，次のようにコスト関数を定義する．

定義 2.1 (コスト関数)
初期頂点 v_0 から頂点 v_i に至る経路を $(v_0, v_1, \cdots, v_i)$ とする．辺 (v_k, v_{k+1}) に対応するコストを辺の重み $w(v_k, v_{k+1})$ としてあらわすとき，コスト関数 $g(v_i)$ は，

$$g(v_i) = \sum_{k=0}^{i-1} w(v_k, v_{k+1}) \tag{2.1}$$

である．

コスト関数 $g(v_i)$ の値は，初期頂点から注目頂点までの経路に沿った辺の重みの総和である（図 2.5）．この値が小さい解ほど優れた解とみなす．コスト最小の経路を最適解と定めるとき，最適解を発見できるかどうかは，探索手法を特徴づける一つの要素である．最

適解の発見を保証している探索手法は，**最適性を有する探索手法**と表現される

コストが確定している状態空間において，初期頂点からすべての頂点に至る経路についてコスト最小の経路を発見する手法は，Dijkstra **のアルゴリズム**として知られている [10]．そこでは効率を高めるために，常にコスト最小の頂点からコストの再計算を行うことが提案されている．

均一コスト探索は，Dijkstra のアルゴリズムにおいて，頂点 v_i に付されるコストをコスト関数 $g(v_i)$ として，その値の昇順に注目頂点を選択し，目標頂点に到達した時点で終了する手法に対応する．

これにより，最適解のコストが $g(v_g)$ であるとき，$g(v_i) > g(v_g)$ となる頂点 v_i は，結果として注目頂点として選択されることなく探索が終了する．これは，コストという新たな情報を利用した探索戦略を用いたことにより，すべての頂点を網羅的に探索することなく，解が発見できるという点で，探索効率が高められたものと解釈できる．

2.3 不確かさのある状態空間における探索手法

問題の性質によっては，探索の開始時点や探索の途中で正確なコストが得られない場合がある．その場合には，問題解決に有用であると考えられる**発見的知識**を利用して推定したコストを利用する探索手法が提案されている．利用する推定コストの特徴によっては，探索効率の向上が期待できる．発見的知識を利用してコストの推定値を算出するヒューリステック関数を次のように定義する．

定義 2.2 (ヒューリスティック関数)
注目頂点 v_i から目標頂点のひとつ v_n に至る経路を $(v_i, v_{i+1}, \cdots, v_n)$ とする．辺 (v_k, v_{k+1}) に対応する推定コストを辺の重み $w(v_k, v_{k+1})$ としてあらわすとき，ヒューリスティック関数 $h(v_i)$ は，

$$h(v_i) = \sum_{k=i}^{n-1} w(v_k, v_{k+1}) \tag{2.2}$$

である．

$h(v_i)$ の値は，注目頂点から目標頂点に到達するまでに要するコストの推定値である（図 2.6）．この値が小さい注目頂点ほど，少ないコストで目標頂点に到達するものと推定できる．

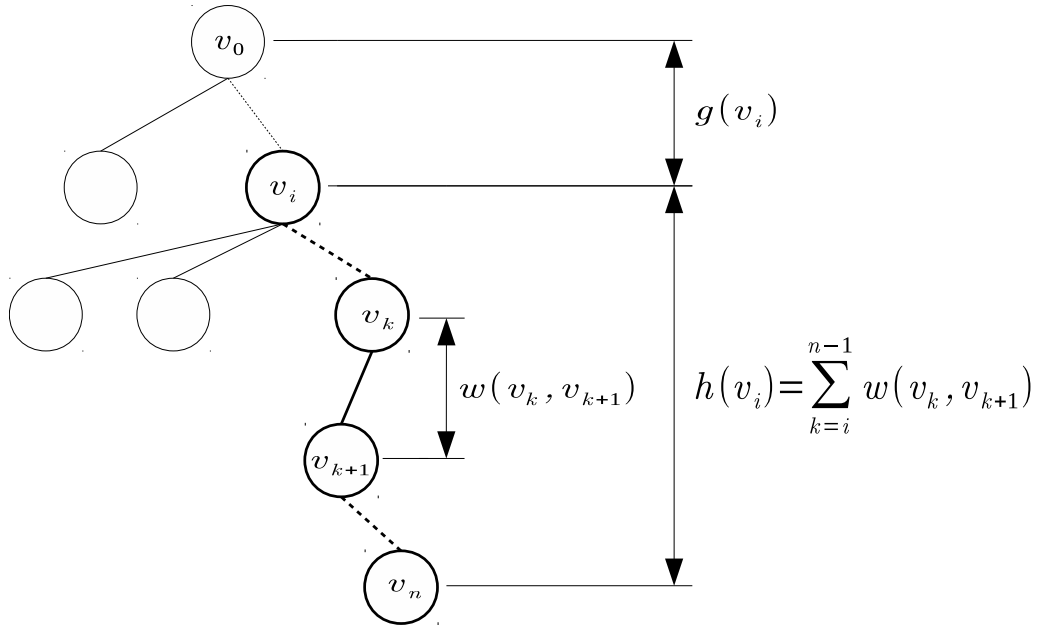


図 2.6　ヒューリスティック関数

探索戦略としてヒューリスティック関数 $h(v_i)$ を含む評価関数 $f(v_i)$ を注目頂点の選択に用いる探索手法は，総称して**最良優先探索**と呼ばれている [2].

貪欲的最良優先探索は，$f(v_i) = h(v_i)$ を評価関数として注目頂点を選択する探索手法である．均一コスト探索と同様に，評価値の小さい頂点から優先的に注目頂点として選択する手法であるが，$h(v_i)$ の値がコストの推定値であるため最適解の発見は保証されない．

評価関数として $f(v_i) = g(v_i) + h(v_i)$ を使用する探索手法は，A **アルゴリズム**と呼ばれている [11]．初期頂点から v_i までのコスト $g(v_i)$ を注目頂点の選択に加味するが，$h(v_i)$ が推定コストであるため最適解の発見は保証されない．

特に，正のコストが割り当てられた有限の状態空間において，ヒューリスティック関数 $h(v_i)$ によって推定されるコストの真値が $h^*(v_i)$ であるとき，すべての頂点について $h(v_i) \leq h^*(v_i)$ をみたす $h(v_i)$ は，**適格性**を有するヒューリスティック関数という．適格性のあるヒューリスティック関数を用いて，コストを評価する A アルゴリズムは，A*アルゴリズムとよばれる．A*アルゴリズムは，最適解の発見が保証される [5].

これに加え，使用するヒューリスティック関数の精度の違いによって生じる探索過程の品質の違いが明確に示されている．具体的には次の性質である．精度の異なる 2 つのヒューリスティック関数 $h_1(v_i)$，$h_2(v_i)$ があり，$h_1(v_i) < h_2(v_i) \leq h^*(v_i)$ を満たし

ているものとする．$f_1(v_i) = g(v_i) + h_1(v_i)$ を評価関数とする A*アルゴリズムを A_1，$f_2(v_i) = g(v_i) + h_2(v_i)$ を評価関数とする A*アルゴリズムを A_2 とすると，A_2 が展開する頂点数は，A_1 が展開する頂点数と同じか少なくなる．このように，A*アルゴリズムでは，ヒューリスティック関数の特性が，発見する解の品質や探索の効率に関わることから，問題の特徴を考慮してヒューリスティック関数に対して工夫がなされる．

解決すべき主問題が複数の副問題に分割できるとき，解決した副問題の解から得られるコストを主問題のヒューリスティック関数の値として用いる**パターンデータベース**という考え方が提案されている [6]．特に適用される作用素も異なるように副問題へ分割できる場合には，それらの解から得られるコストを加えることでより精度の高いヒューリスティック関数にする手法が提案されている [7]．

ヒューリスティック関数を工夫するだけでなく，探索戦略を変更することでも探索過程の品質を高めることが行われている．反復深化深さ制限探索の探索戦略を A*アルゴリズムに導入したものが**反復深化** A***探索**である [9]．これは，反復深化深さ制限探索の経路長に対する制限に変えて，A*アルゴリズムにおいてコストに対する制限を加える．解が発見されるまで，コストに対する制限を緩和しながら，A*アルゴリズムを繰り返し適用するものである．

適格性を損なったヒューリスティック関数を使用した場合の探索の振る舞いについても検討されている．**重み付き** A***探索**は，評価関数を $f(v_i) = g(v_i) + (1 + \alpha)h(v_i)$ とするものである．これにより発見される解のコストは，最適解のコストの $1 + \alpha$ 倍を超えないことが示されている [12]．
また，A*アルゴリズムにおいて適格性を損なったヒューリスティック関数を使用した場合には，最適解の発見を保証できないものの，展開頂点数はより少なくなることが示されている [13]．

不確かさのある状態空間となる問題には，探査を一旦は終えた範囲でもコストが推定値のままとなるものが含まれる．たとえば，探索の実行中にコストが変動するような問題がこれに相当する．このような問題では，初期状態からの経路として発見した解は，問題解決を開始する時点には有効でない可能性がある．そのため，貪欲的最良優先探索のように評価関数を $f(v) = h(v)$ として探査と同時に状態も遷移する方が合理的である．$h(v)$ が適格であることを前提として，コストの変動などにも対応するために，探査済みの頂点の再探査も行い，$h(v)$ の精度を高めながら探索を進める手法として **LRTA*** が提案されている [14]．これは，隣接頂点への移動と $h(v)$ の更新を繰り返しながら，目標頂点を発見

表 2.3　状態空間の分類

種類	探査みの範囲	未探査の範囲
Class0	真値	真値
Class1	真値	推定値
Class2	推定値	推定値

しようとする手法である．1 回の問題解決における最適解の発見は保証されないが，同一状態空間における問題解決を繰り返すことにより，$h(v)$ の精度が高められ，発見する解は最適解に収束する．

LRTA*において経路長 k の範囲のみの近傍探索を行い，その結果を利用して $h(v)$ を更新する手法が **LRTA*(k)** である [15]．LRTA*(k) は LRTA*を一般化した手法である．すなわち，LRTA*(k) において $k = 1$ とした場合が LRTA*になる．

このように，不確かさのある状態空間を対象として，ヒューリスティック関数を使用する探索手法では，探索の品質を向上させるために，発見的知識を導入し精度の高いヒューリスティック関数を構成することや，探索を進める中で獲得できる情報に基づいてヒューリスティック関数を更新することが行われている．

2.4　状態空間と探索手法の特徴

2.4.1　状態空間の分類

状態空間を特徴付けるひとつの要素には，コストの客観性がある．これは，探査によって獲得できる辺のコストを真値であるか，推定値であるかのいずれかに区分するものである．状態空間を探査済みの範囲と未探査の範囲に分割し，それぞれの範囲に含まれる辺のコストの客観性により表 2.3 のように分類する．

Class0　探索を始める前に，コストの真値が状態空間全体に付与される問題に対応する．コストを考慮しない問題もこの種類に含まれるものとする．

Class1　探索を始める前に，発見的知識を利用してコストの推定値を状態空間全体に付与することができ，探査済みの範囲ではコストの真値が獲得できる問題に対応する．探索中にコスト真値が得られない範囲があるという点で，Class0 と比較して解を発

表 2.4　探索手法の特徴

探索戦略	優先順位の決定方法
網羅型	生成順（FIFO）
	生成順（LIFO）
最良優先型	$g(v)$
	$g(v)+h(v)$
	$h(v)$

見するために必要な情報が不足しているとみなせる．

Class2　探索を始める前に，発見的知識を利用してコストの推定値を状態空間全体に付与することができるが，探査済みの範囲においてもコストの真値が獲得できるかどうかわからない問題に対応する．コストの真値が得られるとは限らないという点で，Class1 と比較して解を発見するために必要な情報がさらに不足しているものとみなせる．

2.4.2　探索戦略による探索手法の分類

探索手法を特徴付ける要素は，注目頂点の選択方法としての**探索戦略**と，頂点の優先順位を定める評価関数である．探索戦略は，頂点をもれなく探査する**網羅型**と，優先順位をつけて選択的に頂点を探査する**最良優先型**に大別できる．網羅型の探索戦略では，頂点の生成順に先入れ先出し法，または後入れ先出し法により探査する順序を管理する 2 通りがある．最良優先型の探索戦略では，評価関数により探査順序を決定する．評価関数には，探査済みの範囲と未探査の範囲の境界点として，初期頂点から注目頂点までのコストを評価する $g(v)$ を使用する場合，注目頂点から目標頂点までのコストを評価する $h(v)$ を使用する場合，両者の和 $g(v)+h(v)$ を使用する場合の 3 通りがある（表 2.4）．

Class0 の状態空間において，コストを考慮しない問題の場合には，網羅的に探索することで解を発見することができる．コストを考慮する問題の場合には，$g(v)$ もしくは $g(v)+h(v)$ を評価関数とした最良優先型の探索戦略により，1 回の探索で最適解を発見できる．特に $g(v)+h(v)$ を評価関数とする場合には，事前にすべての経路の真のコストが既知になることから，最適解となる経路以外の頂点を展開することなく問題解決が可能

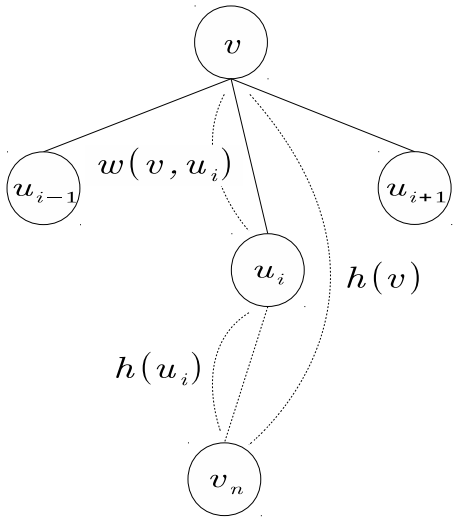


図 2.7　ヒューリスティック関数の更新

である．

Class1 の状態空間においては，未探査の範囲について発見的知識を利用したヒューリスティック関数 $h(v)$ を用いる．$h(v)$ が適格性を有する場合には，$g(v) + h(v)$ を評価関数とした最良優先型の探索戦略により，1 回の探索で最適解を発見できる．すべての頂点について $h(v) = 0$ であるヒューリスティック関数も適格性を有することから，$g(v)$ を評価関数とした最良優先型の探索戦略によっても 1 回の探索で最適解を発見できる．

Class2 の状態空間では，1 回の探索の間にコストの真値を獲得できる保証がないため，発見した解が，コスト最小であるかどうかの判断は困難である．適格性を有するコストの推定値が付与される場合，探索中に v の隣接頂点 u_i を生成した時点で，辺 (v, u_i) のコストの真値 $w(v, u_i)$ が獲得できるものとすると $h(v) \leftarrow \min_i\{w(v, u_i) + h(u_i)\}$ として $h(v)$ を更新することができる（図 2.7）．この結果を用いながら，同一の状態空間における探索を繰り返し行うことで，$h(v)$ が真値に収束することが知られている [14]．同一状態空間での探索を繰り返し実行することが許される問題の場合には，このような探索手法により最終的に最適解を発見することが可能になる．

2.5　探索の品質向上のためのアプローチ

Class0 は，探索前に解を発見するために必要な情報がすべて与えられる状態空間である．探索済みの範囲のコスト $g(v)$ と，未探査の範囲のコスト $h(v)$ のいずれにおいてもコ

ストの真値が得られることから，最良優先型の探索戦略を有し，$f(v) = g(v) + h(v)$ を評価関数とする探索手法により，1 回の探索で最適経路から外れることなく最適解を発見できる．解を発見するために探査するという観点から探索を捉えると，探索を必要としない問題とみなすこともできるが，最も品質の高い探索であると考えることができる．Class0 以外の状態空間における探索において，1 回の探索で最適経路から外れることなく最適解を発見できるよう工夫することが探索の品質向上に必要である．

Class2 は，Class0 とは対照的に，探索前に解を発見するために必要な情報が不足する状態空間である．しかし，Class2 では探査済みの範囲において，コストの真値が得られるとは限らないことを前提にしている．そのため，評価関数にコストの真値ではない $g(v)$ を加えることは，必ずしも探索過程の品質向上には寄与しない．

この状態空間では，発見的知識に基づいてコストを推定するヒューリスティック関数 $h(v)$ を利用した評価関数 $f(v) = h(v)$ の下で，最良優先型の探索戦略により探索する．LRTA(k)*のような探索手法では，$h(v)$ の初期値として適格性を満たす値が付与されることを前提として，探索中に状態空間から局所的に得られる情報に基づいて $h(v)$ を更新しながら探索が進められる．この手法では，1 回の探索の中でも再探査を許容することによって，$h(v)$ の精度が高められる．また，同一の状態空間において直前に得られた $h(v)$ を利用しながら，繰り返し探索を行うことにより，最終的には最適解を導くヒューリスティック関数へと収束する．

これらのことは，適切な初期値から適切な方法で，コストの真値やより精度の高い推定値が得られるように評価関数を更新することにより，探索品質が向上することを示唆するものと考えられる．

これを踏まえ，本研究では，Class1 の木構造で表される状態空間における探索品質の向上のために，問題に合わせて次のアプローチを展開する．

- コストの特徴に応じた発見的知識の下で，探索中に得られる情報に基づき，広範囲にわたってヒューリスティック関数を更新する．
- 探索の過程で部分的に獲得される情報に基づきコスト関数を更新する．

Class1 の状態空間は，探査済みの範囲についてコストの真値が得られる．最良優先型の探索戦略の下で，コストの真値 $g(v)$ と適格性のあるヒューリスティック関数 $h(v)$ で構成される評価関数 $f(v) = g(v) + h(v)$ を用いる探索手法により，1 回の探索で最適解を発見

できる．

前者のアプローチは，状態空間に付されたコストの特徴が既知である問題に適用する．その特徴を反映した発見的知識に基づいてひとつの頂点の探査結果を複数の頂点を対象としたヒューリスティック関数の更新に利用するものである．これによりヒューリスティック関数の精度が向上し，解の発見までの探査範囲の縮小が期待できる．また，適格性を損なわないようにヒューリスティック関数を更新することで，最適解の発見を保証することが期待できる．

後者のアプローチは，探索中に得られる情報が部分的であるために，探査済みの範囲におけるコストの真値を得るための再探査が必要となるような問題に適用する．ある状態への 1 回の探査でコストの真値が得られない場合には，最適解の発見を保証することが困難になる．また，同一状態空間における繰り返しの探索も許容できない場合には，ヒューリスティック関数の収束を待つことができない．そこで，情報の追加的な収集を行い，獲得した情報を利用したコスト関数の更新と再探査により，複数回の探索を繰り返すことなく最適解の発見を保証しようとするものである．

第 3 章

コストの変動を考慮した探索手法に関する一考察

3.1 はじめに

自動車や鉄道を利用した移動では，道路の渋滞や列車の遅延などの障害に起因して，目的地までの移動時間が通常時の見込みから増加することがある．交通網の発達した地域では，目的地までの移動経路や移動手段を複数確保できる場合があり，それらを適切に選択すると，移動時間の増加を最小限にとどめられる可能性がある．このような例の他にも，周囲の状況が変化する中で，その変動を考慮しながら，より望ましい代替案を選択しようとする活動は日常的にも多くみられる．

本章では，交通機関を利用して移動する場合の移動時間のように，状況変化によって変動する要素をコストとみなし，コストが変動する状況下における代替案の決定手法としての探索手法について検討する．

交通網や鉄道網などを想定すれば，迂回路や並行路線などが相互に接続する接合点に到達したとき，どの経路を移動するかを選択する機会が得られる．選択後の移動中においても状況は変化するが，次の接合点に到達するまで移動に要した時間は確定しない．ある経路を選択して一旦移動を始めると，通常はその選択を行った地点に瞬時に戻るような取り消しは容易ではない．したがって，到達した接合点における選択肢は，そこに到達する以前に選択した経路によって限定され，それまでに現れた選択肢を対象にはできない [16]．また，選択しなかった経路について，どのぐらいの時間で移動可能であったかを正確に知る方法はない．このような状況に対応する状態空間は，探査済みの範囲においてもコスト

が推定値にとどまり，探索が段階的に部分木へと進行し，生成済みの上位の頂点に戻ることがない木構造となる．そのため，生成済みの頂点を任意の時点で選択しなおせることを前提としない最適解を定義する必要がある [17].

そこで，まず，注目頂点の選択時点のコストに基づいて，評価の高い頂点から順に最良優先探索を行った場合に最終的に到達した解を最適解と定める．このような最適解を発見するために，ヒューリスティック関数を注目頂点から隣接頂点までの短期予測と，隣接頂点から目標頂点までの長期予測で構成した探索手法を提案する．

次に，提案手法の特性のひとつとして，ここでの定義に基づく最適解への到達性について検討を加え，コストの予測に対する条件を示す．

最後に，短期予測と長期予測の予測精度のバランスについて考察し，両者の精度の相対的な高低関係による予測能力について定性的に言及する．

本考察の結果，コストの予測については，A*アルゴリズムの適格性の条件と同様に，真のコストを超えない予測の必要性が示される．また，予測精度と予測能力との関係については，真の値と予測値との誤差を最小にするという点からの予測精度以外にも，短期予測と長期予測の 2 種類の予測の精度を揃えることの重要性が示される．

3.2　コスト変動下における探索

3.2.1　問題設定

状態空間は有限の木構造である．頂点 v，w の間の辺 (v, w) には正のコスト $C(v, w)$ が付される．その初期値 $C_0(v, w)$ は既知とする．注目頂点を選択した時点で探索が 1 段階進行したものとみなし，その段階に合わせて各辺のコストが変動するものとする．その変動量は非負であり，状態空間全体でコストは単調に増加するものとする．

各頂点において，その頂点 w を根とする部分木 $T_s(w)$ が定義できる．部分木 $T_s(w)$ の最小コスト $h(w)$ は，部分木内の経路のコストのうちの最小値とする．コストの初期値に基づいた場合の最小コストを部分木の最小コストの初期値とする．

3.2.2　最適経路と最適解

探索の実行中にコストが変動するため，最適解は変化する可能性があり，それに応じて最適経路も変化する可能性がある．コスト変動下における最適経路と最適解を次のように

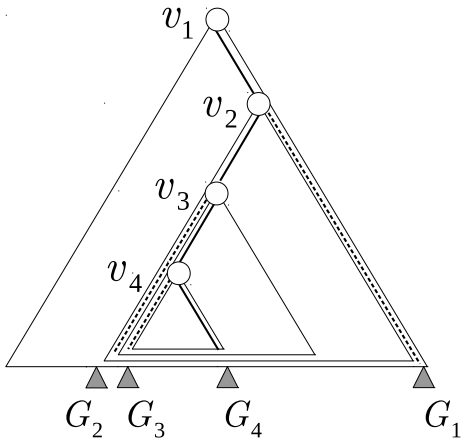


図 3.1　最適解に至る経路

定める．

探索の第 i 段階において v_i を根とする部分木 $T_s(v_i)$ 内のその時点のコストに基づくコスト最小の経路が $p(v_i) = (v_i, v_{1+1}, v_{i+2}, \cdots, G_i)$ であるとする．次の第 $i+1$ 段階では，$p(v_i)$ 上の頂点 v_{i+1} を注目頂点として選択し，部分木 $T_s(v_{i+1})$ 内のその時点のコストに基づくコスト最小の経路 $p(v_{i+1}) = (v_{i+1}, v_{i+2}, \cdots, G_{i+1})$ を求める．これを初期頂点 S から再帰的に繰り返して探索が終了したときに到達する頂点 G_n を最適解とする．また，この過程で経由した頂点の列 $(S, v_2, v_3, \cdots v_{n-1}, G_n)$ を最適経路とする．

具体例を図 3.1 に示す．$v_1, \cdots, v_4$ のそれぞれを根とする部分木にけるコスト最小の経路により到達する頂点が，それぞれ $G_1, \cdots, G_4$ であるとする．第 1 段階では，v_1 から G_1 に至る経路 $p(v_1) = (v_1, v_2, \cdots, G_1)$ がコスト最小の経路である．. 第 2 段階では，v_2 を選択され，コスト最小の経路 $p(v_2) = (v_2, v_3, v_4, G_2)$ が得られる以下，第 3 段階で $p(v_3) = (v_3, v_4, G_3)$，第 4 段階で $p(v_4) = (v_4, G_4)$ となり，最適解が G_4，最適経路が $(v_1, v_2, v_3, v_4, G_4)$ である．

3.2.3　探索手法の概要

評価関数

図 3.2 のように頂点 v_i から生成した隣接頂点を $w_1, \cdots, w_j, \cdots, w_m$ とする．v_i，w_j 間の辺のコストの予測値を $\hat{C}(v_i, w_j)$，部分木 $T_s(w_j)$ の最小コストの予測値を $\hat{h}(w_j)$ とする．頂点 v_i から w_j を経由する経路のコストを評価する評価関数 $\hat{f}(w_j)$ を次のように

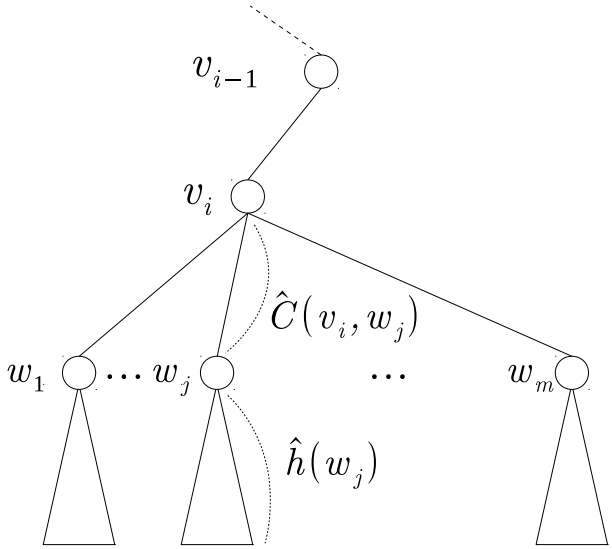


図 3.2　辺と部分木のコスト

定義する．

$$\hat{f}(w_j) = \hat{C}(v_i, w_j) + \hat{h}(w_j) \tag{3.1}$$

$f(x)$ の最小値を与える引数 x が返される関数を $\mathrm{argmin}_x\{f(x)\}$ とし，次に注目頂点として選択する頂点 v_{i+1} を，評価関数 $f(w_j)$ に基づいて次式で定める．このように経路の評価と頂点の選択を繰り返して探索を進める．

$$v_{i+1} = \underset{j}{\mathrm{argmin}}\{\hat{C}(v_i, w_j) + \hat{h}(w_j)\} \tag{3.2}$$

短期予測

注目頂点 v_i と隣接頂点 w_j の辺 (v_i, w_j) におけるコストの予測を**短期予測**と呼ぶことにする．注目頂点として v_i を選択した時点で，その時点の辺 (v_{i-1}, v_i) のコストの真値 $C(v_{i-1}, v_i)$ を観測することができる．辺 (v_{i-1}, v_i) のコストの初期値 $C_0(v_{i-1}, v_i)$ から $C(v_{i-1}, v_i)$ への変動と同程度の変動が，辺 (v_i, w_j) のコストの初期値 $C_0(v_i, w_j)$ に対して生じるものと仮定する．辺 (v_i, w_j) に対する短期予測 $\hat{C}(v_i, w_j)$ を次のように定義する．

$$\hat{C}(v_i, w_j) = C_0(v_i, w_j) \cdot \frac{C(v_{i-1}, v_i)}{C_0(v_{i-1}, v_i)} \tag{3.3}$$

長期予測

頂点 w_j の部分木 $T_s(w_j)$ における最小コストの予測を**長期予測**と呼ぶことにする．初期頂点 S から注目頂点として選択された頂点 v_i までの経路 $p = (S, v_2, \cdots, v_i)$ について，その時点のコストの真値 $g(v_i)$ が得られる．経路 p のコストの初期値 $g_0(v_i)$ から $g(v_i)$ への変動と同程度の変動が，部分木 $T_s(w_j)$ の最小コストの初期値 $h_0(w_j)$ に対して生じるものと仮定する．部分木 $T_s(w_j)$ の最小コストに対する長期予測 $\hat{h}(w_j)$ を次のように定義する．

$$\hat{h}(w_j) = h_0(w_j)\frac{g(v_i)}{g_0(v_i)} \tag{3.4}$$

ここで $g_0(v_i)$，$g(v_i)$ は次のように定めるものとする．なお，v_1 は初期頂点 S とみなす．

$$g_0(v_i) = \sum_{k=1}^{i-1} C_0(v_k, v_{k+1}) \tag{3.5}$$

$$g(v_i) = \sum_{k=1}^{i-1} C(v_k, v_{k+1}) \tag{3.6}$$

$$\tag{3.7}$$

3.3 探索手法の特性

最適経路に沿って探索が進む可能性を検討する．図 3.3 に頂点 v の部分木 $T_s(v)$ が展開される様子を示す．隣接頂点 w の部分木 $T_s(w)$ に，この時点のコストに基づく v からの最適解 G が含まれるものとする．同様に w' の部分木 $T_s(w')$ には，この時点の v からの準最適解 G' が含まれるものとする．v から w，w' へのコストの真値をそれぞれ $C(v, w)$，$C(v, w')$ とする．また，部分木 $T_s(w)$，$T_s(w')$ の最小コストの真値をそれぞれ $h(w)$，$h(w')$ とする．

注目頂点を選択する時点では，コストの真値を知ることはできないため，推定値により評価する．任意の頂点を x，x の任意の隣接頂点が y であるとき，次の条件を満たすようにコストを予測できるものとする．

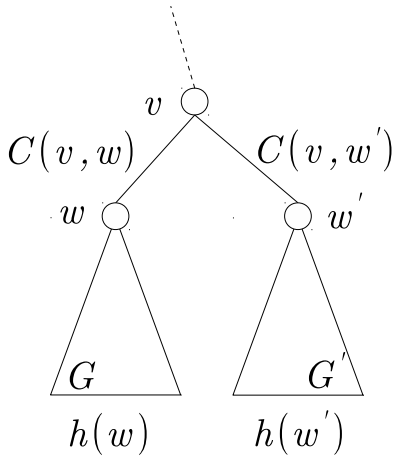


図 3.3　部分木と最適経路

$$\hat{C}(x,y) \leq C(x,y) \tag{3.8}$$

$$\hat{h}(y) \leq h(y) \tag{3.9}$$

式 (3.1) に基づき，頂点 y を経由する経路のコストの推定値を $\hat{f}(y)$，その真値を $f(y)$ とする．w を経由して G に至る経路のコストについて，式 (3.8)，式 (3.9) の条件のもとで次式が成り立つ．

$$\hat{f}(w) = \hat{C}(v,w) + \hat{h}(w) \leq C(v,w) + h(w) = f(w) \tag{3.10}$$

$f(w)$ は v からこの時点の最適解 G への経路のコストの真値であるから，v の部分木 $T_s(v)$ の最小コスト $h(v)$ に等しい．

$$\hat{f}(w) \leq f(w) = h(v) \tag{3.11}$$

一方，w' を経由する経路は，v から準最適解への経路である．そのコストは $h(v)$ より大きい．

$$f(w') > h(v) \tag{3.12}$$

3.3.1　解の移動がない場合

ここで，w' を選択し，以後，コスト変動が生じても解が移動することなく G' に到達し，探索を終了するものと仮定する．OPEN リストに w を保持していたとすると，コスト変

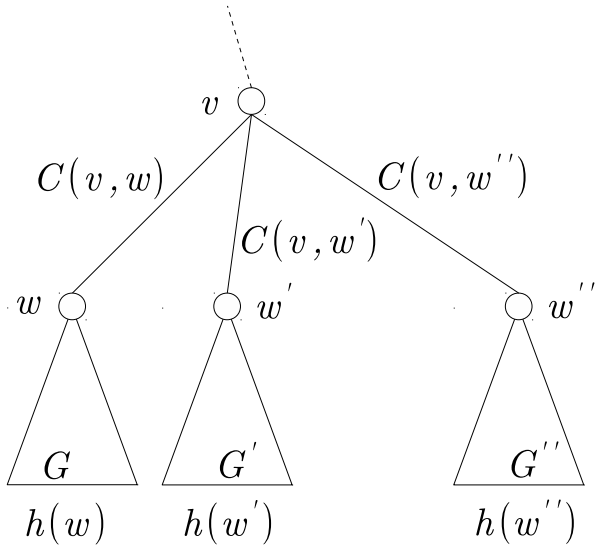


図 3.4 解の移動がある場合（1）

動によって $\hat{f}(w)$ を超えた時に G' へ到達できなくなる．そのため，この仮定が成立するためには，探索終了時に既知となる $f(w')$ が次式を満たさなければならない．

$$f(w') < \hat{f}(w) \tag{3.13}$$

式 (3.12)，式 (3.13) から次式が導かれる．

$$h(v) < f(w') < \hat{f}(w) \tag{3.14}$$

これは式 (3.11) と矛盾する．したがって，式 (3.8)，式 (3.9) の条件のもとでコストを予測し，OPEN リストを持てば，その時点の最適解に向かう経路上の w があるにも関わらず，別の経路をたどって探索を終了することはない．

3.3.2 解の移動がある場合

コスト変動により解が G' から G'' へ移動する時，G'' が部分木 $T_s(w')$ からはずれない場合は，G'' を G' に置き換えることができ，前述の解の移動がない場合と同じである．また，G'' が部分木 $T_s(w')$ からはずれる場合でも，G のない部分木に移るならば，コスト変動後の状況は，前述の解の移動がない場合と同じである（図 3.4）．

したがって，G のある部分木 $T_s(w)$ に G'' が移る場合を考慮する（図 3.5）．G'' に到達し，探索を終了するものと仮定すると次式を満たさなければならない．

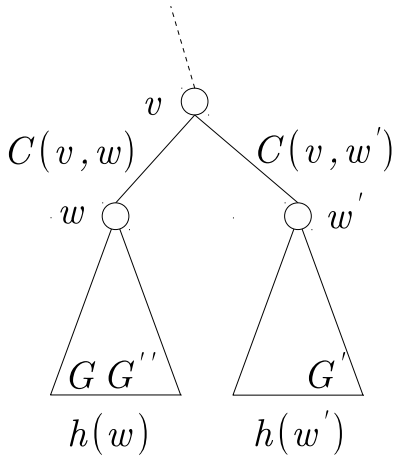


図 3.5　解の移動がある場合（2）

$$f(G'') < \hat{f}(G) \tag{3.15}$$

ここで,

$$f(G'') = C(w, G'') + h(G'') \tag{3.16}$$
$$\hat{f}(G) = \hat{C}(w, G) + \hat{h}(G) \tag{3.17}$$
$$h(G'') = \hat{h}(G) = 0 \tag{3.18}$$

であるから,

$$C(w, G'') < \hat{C}(w, G) \tag{3.19}$$

式 (3.8) の仮定から

$$C(w, G'') < \hat{C}(w, G) \leq C(w, G) \tag{3.20}$$

となり，G が最適解であることに反するため，準最適解を選択して探索を終了することはない.

したがって，式 (3.8)，式 (3.9) の条件のもとでコストを予測し，OPEN リストを持てば最適解を発見せずに探索を終了することはない.

3.3.3　最適解への到達性評価

本手法が最適解に到達する能力を持つためには，式 (3.8)，式 (3.9) を満たす予測が可能であり，かつ，OPEN リストを持つことである．しかし，本章で対象とする問題は，前段

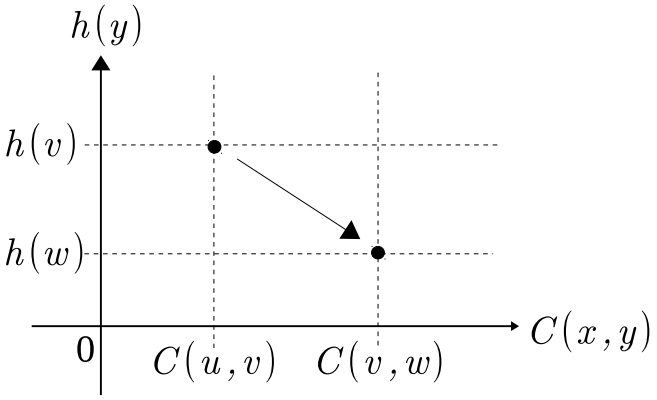


図 3.6　経路平面

階の決定の下で，以後の選択肢が限定されるという性質を有するため，選択されなかった頂点を OPEN リストに保持することはしない．そのため，探索のある時点で最適解へ至る頂点が選択されなければ，その時点で本手法は最終的な最適解への到達性を失う．したがって，式 (3.8)，式 (3.9) を満たしつつ，可能な限り最適経路と非最適経路のコストの大小関係を保った予測を行うことが重要である．

3.4　予測精度と状態空間との関連

コストの予測精度によって，最適経路と非最適経路のコストの大小関係を保った予測（以下，これを正しい予測と表現する）がどの程度可能になるかを検討する．

3.4.1　経路平面

横軸に辺 (x, y) のコスト $C(x, y)$，縦軸に部分木 $T_s(y)$ の最小コスト $h(y)$ をとり，状態空間に存在する経路のコストを座標 $(C(x, y), h(y))$ で表現する経路平面を考える（図 3.6）[18].

例えば，図 3.7 のような木構造で，頂点 u，v，w を経由する経路を探索の各段階で選択したとすると，図 3.6 の矢印で示すように経路平面上を移動したことになる．

頂点 v から w を経由する最適経路と w' を経由する非最適経路のコストの真値について，$C(v, w) + h(w) < C(v, w') + h(w')$ から，次の関係が成立する．

$$h(w') - h(w) > -(C(v, w') - C(v, w)) \tag{3.21}$$

$(C(v, w'), h(w'))$ の存在範囲，すなわち，非最適経路の存在範囲は，経路平面で

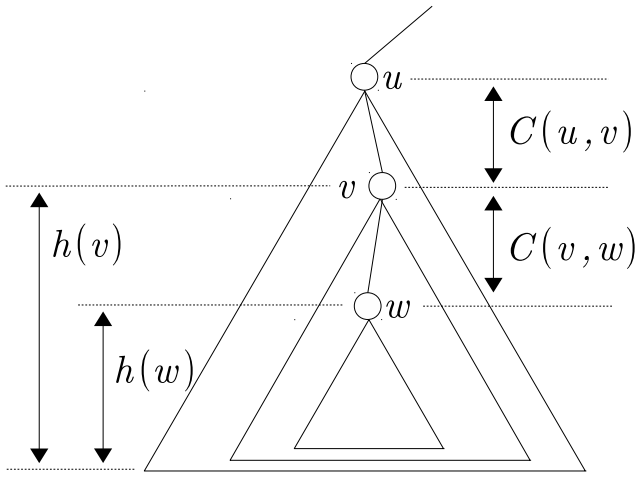


図 3.7　経路平面に現れる経路の例

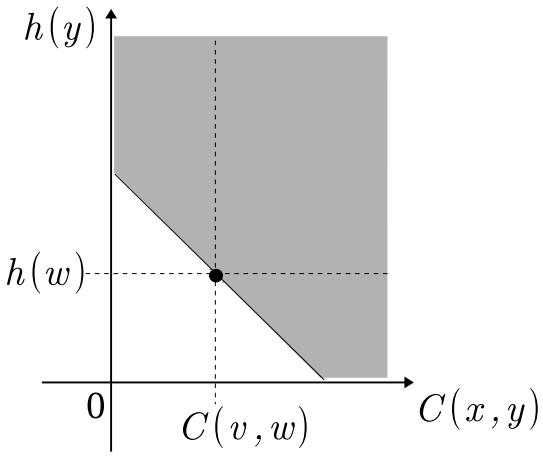


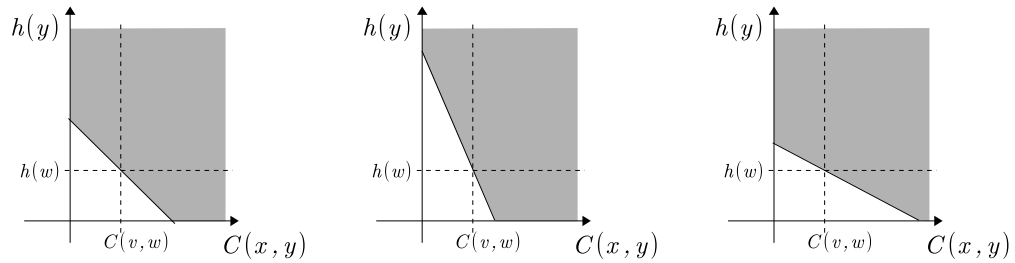
図 3.8　経路平面での非最適経路の存在範囲

$(C(v,w),h(w))$ を通り，傾きが -1 の直線の上側の領域である．これを図 3.8 に示す．

3.4.2　予測精度の特徴

$C(x,y)$，$h(y)$ の予測精度をそれぞれ $0<\alpha\leq 1$，$0<\beta\leq 1$ とする．予測値はそれぞれ，$\hat{C}(x,y)=\alpha C(x,y)$，$\hat{h}(y)=\beta h(y)$ と表すことができる．頂点 v からの経路について，正しい予測ができる場合には，$\alpha C(v,w)+\beta h(w)<\alpha C(v,w')+\beta h(w')$ であるから，次の関係が成立する．

$$h(w')-h(w)>-\frac{\alpha}{\beta}(C(v,w')-C(v,w)) \tag{3.22}$$



（a）$\alpha/\beta = 1$　　（b）$\alpha/\beta > 1$　　（c）$\alpha/\beta < 1$

図 3.9　予測精度の特徴と非最適経路の存在範囲

これを満たす頂点 v からの非最適経路 $(C(v,w'), h(w'))$ の存在範囲は，$(C(v,w), h(w))$ を通る直線の上側の領域である．境界となる直線の傾きの絶対値は予測精度の比に対応する．α，β の相対的な大小関係に基づいて，予測精度の特徴を次のように分類する．

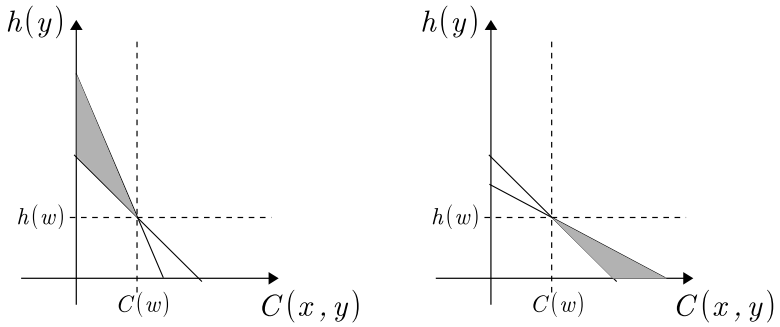
(a). $C(x,y)$，$h(y)$ の予測精度が同程度である（$\alpha/\beta = 1$）
(b). $C(x,y)$ の予測精度が $h(y)$ の予測精度より相対的に高い（$\alpha/\beta > 1$）
(c). $C(x,y)$ の予測精度が $h(y)$ の予測精度より相対的に低い（$\alpha/\beta < 1$）

上記の予測精度の特徴（a）～（c）の下で，その予測値に基づいて非最適経路となる経路の存在範囲の例を図 3.9（a）～（c）にそれぞれ示す．

3.4.3　予測精度の特徴と予測の正しさ

図 3.9（a）に示すように，$C(x,y)$，$h(y)$ の予測精度が同程度である場合には，予測値によって非最適経路として扱われる経路の存在範囲が，図 3.8 に示された非最適経路の存在範囲とほぼ一致する．これは，本来の非最適経路のすべてをほぼ正しく予測できることを意味するものと考えられる．また，予測精度は同程度であればよいので，その良し悪しとは無関係であることが分かる（なお，$\alpha = \beta$ である場合には，存在範囲は完全に一致し，すべて正しく予測できる）．

$C(x,y)$ の予測精度が $h(y)$ の予測精度より相対的に高い場合は，図 3.9（b）に示すように，図 3.9（a）と比較して境界線の傾斜が大きくなる．α，β の差が大きくなるにつれて，この傾斜は大きくなる．これにより，非最適経路として予測できない経路が生じる．図 3.10（a）に示す範囲に存在する経路がそれである．この場合，本来，非最適経路である経路のうち，短期予測の値が小さい経路の一部をそれとして予測できないことになる．



（a）$\alpha/\beta > 1$　　（b）$\alpha/\beta < 1$

図 3.10　予測できない非最適経路の存在範囲

一方，$C(x, y)$ の予測精度が $h(y)$ の予測精度より相対的に低い場合は，図 3.9（c）に示すように，図 3.9（a）と比較して境界線の傾斜が小さくなる．α，β の差が大きくなるにつれて，この傾斜は小さくなる．これにより，前述の場合と同様に，非最適経路として予測できない経路が生じる．図 3.10（b）に示す範囲に存在する経路がそれである．この場合，長期予測の値が小さい経路の一部をそれとして予測できないことになる．

以上から，本来，非最適経路である経路をそれとして正しく予測するためには，$C(x, y)$，$h(y)$ のそれぞれの予測精度を可能な限り揃えることが重要となることがわかる．

3.5　関連研究

状態空間や探索手法には，それを特徴付けるいくつかの観点がある [19]．その観点に基づけば，本章ではコストの時間的な変動が生じる動的環境を対象とし，実時間探索に属す手法の適用を前提とするものである．実時間探索の代表的な手法に RTA*がある [14]．そこでは，隣接頂点への移動コストと，その頂点からゴールへの推定コストの和で評価する．その評価値で移動前の状態の推定コストを更新する．本章で短期予測と長期予測の和で経路を評価するのは，これに準じたものである．また，RTA*に類似する推定コストの更新方法とダイクストラ法を組み合わせて，渋滞回避を目的とした経路探索手法が提案されている [20]．これは，短期予測で渋滞情報を利用し，推定コストの更新により長期予測の精度を高めるものとみなせる．本章では，コストの更新方法の検討が課題として残されている．

3.6 おわりに

コストの変動を考慮した探索手法について検討した．RTA*などの実時間探索の手法と同様に，最適解への到達を保証するものではないが，最適解への到達可能性を前提として，コストの予測値が満たすべき条件を提示した．コストの予測は短期予測と長期予測の 2 種類で行うが，真のコストを超えないという条件の下で，両者の予測精度を揃えることが予測誤りの減少に寄与するであろうことを示した．

提案する探索手法の予測コストの算出方法の検証や予測値や部分的に得られる観測コストによる更新，および状態空間でのコスト変動の反映の方法など，多くの検討事項が今後の課題として残されている．

第 4 章

経路の特徴を反映した評価値の更新を伴う木探索

4.1 はじめに

発見的探索法は，対象となる問題に対する経験的知識を反映した評価関数を利用して，探索効率を向上させる手法である [5]．経路探索問題では，目的地までの距離や所要時間などをコストとみなし，最短経路としてコスト最小の経路を求めることが行われる．探索開始前には実際のコストは未知であるから，探索はその推定値を利用して進められる．代表的な手法である A*アルゴリズムでは，推定値が実際のコスト以下であり，より実際のコストに近いほど探索効率の向上が期待できる [21]．

問題の性質は，状態空間におけるコストの分布の特徴として現れることがある．たとえば，道路網では交通量が増加すると，移動距離あたりの所要時間は長くなり，結果として渋滞が発生する [22]．鉄道網では乗客の乗降に想定以上の時間を要して遅延が一旦発生すると，その影響は後続列車におよび，次第に拡大する [23]．一旦渋滞や遅延が発生すると，その影響により新たな渋滞や遅延の起点が生じ，周辺地域や関連路線に拡大することもある [24]．このとき，渋滞や遅延の起点に近い区間ほど遅延時間が長い状況に陥っている．これらの状態は，所要時間や遅延時間を区間に付与されるコストと捉えると，その渋滞や遅延の起点に近づくほど，実際のコストが大きくなるという特徴のある状態空間とみなせる．渋滞や遅延の程度は状況によって異なるため，所与のコストが必ずしも有効ではない場合も生じる [25]．また，自然環境においては，降雨により土壌に蓄えられる土壌水分量に応じて土砂災害の危険が次第に高まることが知られている [26]．降雨が継続する山

間の地域内を目的地に向かって移動する場合，時間の経過とともに選択可能などの経路も次第に危険度が高まるため，相対的により安全な経路を計画する必要がある．このような状況での危険度を道路のコストと捉えると，目的地に向かってコストが大きくなる状態空間とみなせる．

本章では，このような性質のある問題においては，コストの特徴を反映するように推定値を更新しながら探索すると，探索効率が高められる可能性があることを示すものである．この問題に適した探索手法として，A*アルゴリズムに推定値の更新方法を埋め込んだ探索手法を提案する [27].

以下では，まず，対象とする状態空間におけるコストの特徴と，推定値の更新方法を示す．状態空間は木構造であり，根から葉に向かう経路にそって，各辺には順次大きいコストを付与されるものとする．渋滞や遅延が生じ始めた道路網や鉄道網，継続する悪天候による災害の危険度の増加など長期的には回復が見込まれるものの，短期的にはこのような状態空間としてモデル化できるものと考えられる．コストの推定値は，直近に観測されるコストの真値を利用して更新する．

次に，探索手法を示し，対象とする状態空間での探索時に現れる特性について述べる．探索手法は A*アルゴリズムにコストの推定値を更新する機能を組み合わせたものである．そこでは，探索手法が最適解を発見することと，解を発見するまでの展開頂点数がコストの推定値を更新しない場合と同じであるか，または少なくなることを理論的に保証する．

最後に，完全二分木を対象としたシミュレーションを用いて，初期状態において精度の高いコストの推定値が与えられた場合でも，A*アルゴリズムより狭い探索範囲で解が発見されることを実験的に示す．以上から，コストの分布に特徴がある状態空間において，評価関数の更新を伴う探索手法は，探索の品質向上に有効な手法であると考えられる．

4.2　関連研究

発見的探索法は，木構造やグラフ構造で表現される状態空間において，求解に有効な発見的知識をコストの評価関数に反映させ，効率的に解を発見しようとする手法である．代表的な手法のひとつに A*アルゴリズムがある．対象となる状態空間が木構造として表されるとき，現在の状態 x から目標状態に至る経路の推定コスト $h(x)$ が真のコスト $h^*(x)$ を超えない範囲で精度が高いほど効率よく最適解を発見することができる．対象となる状態空間には，通常，推定コストが事前に割り当てられるが，真のコストと大きく異なるこ

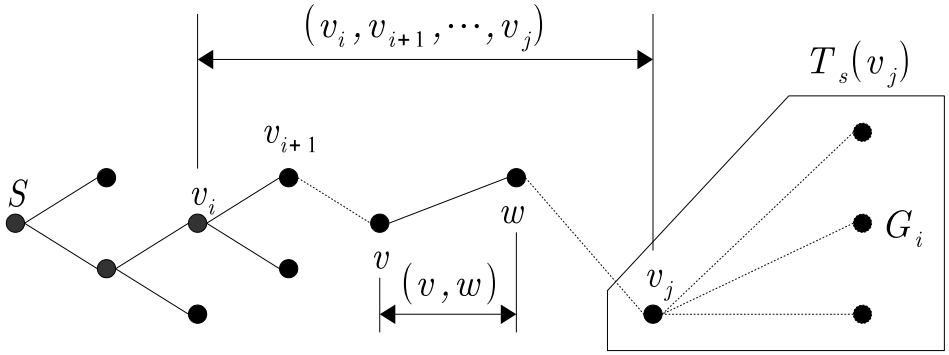


図 4.1 状態空間の表現

とがある．このような場合に，推定コストを更新しながら探索する手法として，実時間探索の代表的な手法である RTA*や LRTA*[14] がある．LRTA*は現在の状態 x から隣接する状態のひとつである y への状態遷移で観測された推定コスト $h(y)$ を用いて，もとの状態 x の推定コスト $h(x)$ を更新する．LRTA*(k)[15] や LRTA*$_{LS}(k)$[28] は，現在の状態 x から次に遷移する状態 y の決定に先立ち，$h(x)$ とそれまでの経路に現れている状態 p_i における推定コスト $h(p_i)$ を更新する．LRTA*と同様に x や p_i の隣接状態のひとつへ移動し，そこで観測された推定コストを利用して $h(x)$ や $h(p_i)$ を更新する．

LRTA*や LRTA*(k) のような実時間探索では，推定コストの更新を伴う複数回の同一状態への遷移を繰り返す．これにより，各状態の推定コストが真のコストに収束し最適解に到達する．一方，A*アルゴリズムでは，推定コストが適格性を満たす限り最適解に到達する．ただし，推定コストの精度が低い場合には探索範囲が広くなる．扱う問題の性質によっては，推定コストが真のコストに収束するまでに要する時間や，広範囲を探索するために要する時間を許容できない場合がある．そのため，同一状態への遷移を回避しつつ推定コストの精度を高めながら，解を発見する手法が望まれる．

4.3 提案手法

4.3.1 状態空間

状態空間は図 4.1 に示すような木構造とする．ある頂点 v の隣接頂点のひとつが w であるとき，この間の辺を (v, w) で表す．特に，初期頂点を S，葉を G_i $(i = 1, \ldots, n)$ と表記する．また，v を根とみなした部分木を $T_s(v)$ で表す．辺 $(v_i, v_{i+1}), (v_{i+1}, v_{i+2}), \cdots$ を経由して，v_i から到達可能な頂点 v_j へ至る経路は，$(v_i, v_{i+1}, \cdots, v_j)$ で表す．このよ

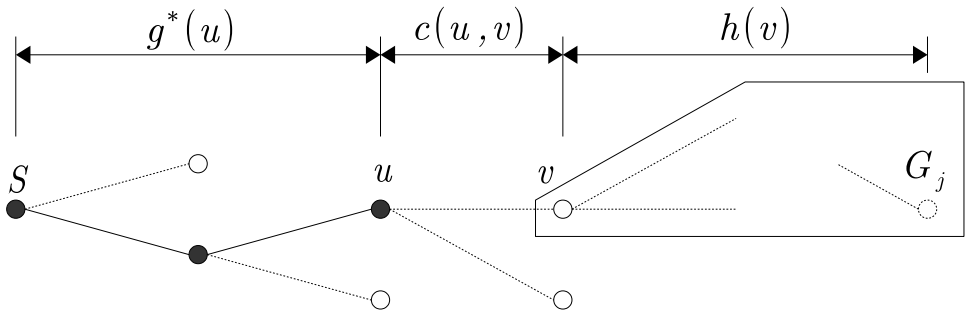


図 4.2　評価関数の構成

うな経路において，頂点 v_i を頂点 v_{i+1} の親頂点と呼ぶ．頂点間の各辺には正の値を持つコストが付与される．辺 (v,w) の真のコストを $c^*(v,w)$ とし，その推定値（以後，推定コストと記す）を $c(v,w)$ で表す．また，推定コストの所与の値（以後，初期値と記す）を $c_0(v,w)$ とする．対象とする木構造について，次の前提と仮定をおく．

前提 4.1 (コストの分布)

初期頂点から葉に向かう経路に沿って，辺の真のコストは非減少である．すなわち，頂点 u とその隣接頂点 v，さらに v の隣接頂点 w からなる経路 (u,v,w) の 2 つの構成辺 (u,v)，(v,w) の真のコストについて

$$c^*(u,v) \leq c^*(v,w) \tag{4.1}$$

である．

仮定 4.1 (推定コストの初期値)

初期状態では，各辺の推定コストの初期値が既知であり，それは真のコストを超えないものとする．すなわち，辺 (v,w) について

$$c_0(v,w) \leq c^*(v,w) \tag{4.2}$$

である．

4.3.2　評価関数

状態空間において探索を進める上で，次に展開する頂点を選択するための評価関数を定める．

定義 4.1 (経路の真のコスト)

経路 $(v_i, v_{i+1}, \cdots, v_j)$ の真のコスト $p^*(v_i, v_j)$ は，その経路を構成する各辺の真のコスト

の和である．

$$p^*(v_i, v_j) = \sum_{k=i}^{j-1} c^*(v_k, v_{k+1}) \tag{4.3}$$

特に，初期頂点 S から v_i に至る経路 $(S, v_1, \cdots, v_i)$ の真のコスト $p^*(S, v_i)$ を $g^*(v_i)$ と表記する．すなわち，$g^*(v_i) = p^*(S, v_i)$ である．

定義 4.2 (経路の推定コスト)

経路 $(v_i, v_{i+1}, \cdots, v_j)$ の推定コスト $p(v_i, v_j)$ は，その経路を構成する各辺の推定コストの和である．

$$p(v_i, v_j) = \sum_{k=i}^{j-1} c(v_k, v_{k+1}) \tag{4.4}$$

特に，定義 4.2 において推定コスト $c(v_k, v_{k+1})$ をその初期値 $c_0(v_k, v_{k+1})$ で置き換えて得られる値を経路の推定コストの初期値 $p_0(v_i, v_j)$ とする．

定義 4.3 (目標コスト)

頂点 v から葉への目標コスト $h^*(v)$ は，v から到達可能な葉 $\{G_1, \cdots, G_m\}$ へ至る各経路の真のコスト $p^*(v, G_1), \cdots, p^*(v, G_m)$ の最小値である．

$$h^*(v) = \min_{k=1}^{m} \{p^*(v, G_k)\} \tag{4.5}$$

葉 $\{G_1, \cdots, G_m\}$ の中に最適解である目標頂点 G_{opt} が含まれるとき，$h^*(v)$ は最適解の目標コストである．

定義 4.4 (推定目標コスト)

頂点 v から葉への推定目標コスト $h(v)$ は，v から到達可能な葉 $\{G_1, \cdots, G_m\}$ へ至る各経路の推定コスト $p(v, G_1), \cdots, p(v, G_m)$ の最小値である．

$$h(v) = \min_{k=1}^{m} \{p(v, G_k)\} \tag{4.6}$$

特に，定義 4.4 において $p(v, G_k)$ をその初期値 $p_0(v, G_k)$ に置き換えて得られる値を推定目標コストの初期値 $h_0(v)$ とする．

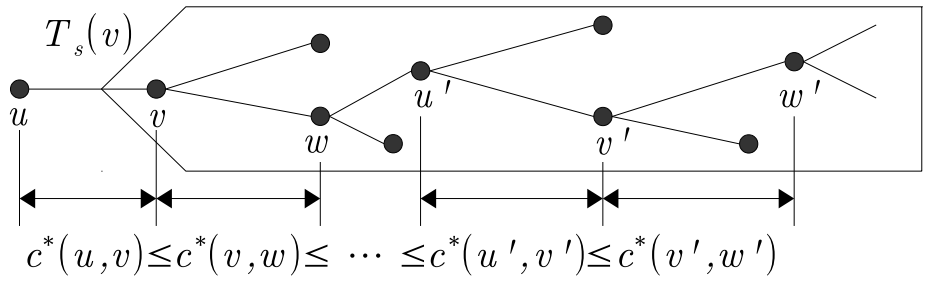


図 4.3　経路に沿った各辺の真のコストの関係

定義 4.5 (評価関数)

初期頂点 S から到達した途中の頂点 u とその隣接頂点のひとつ v を経由し，v から到達可能な葉のひとつ G_j に至る経路 $(S, \cdots, u, v, \cdots, G_j)$ の評価関数を $f(u)$ とする．$f(u)$ は，S から u に至る経路の真のコスト $g^*(u)$，辺 (u, v) の推定コスト $c(u, v)$，v からの推定目標コスト $h(v)$ の 3 項の和である．

$$f(u) = g^*(u) + c(u, v) + h(v) \tag{4.7}$$

図 4.2 は u が展開された時点の状態空間の様子と u の隣接頂点のひとつ v に着目した場合の評価関数 $f(u)$ の各項との対応を表した模式図である．展開した頂点を黒丸で記し，生成した頂点は白丸で記す．破線で記した頂点は未生成の頂点である．実線で表した辺は，その辺を含む経路を評価するときに，真のコストで評価される部分であり，破線で表した辺は，推定コストで評価される部分である．v を含む矩形領域は部分木 $T_s(v)$ である．

4.3.3　推定コストの更新

頂点 u を展開したときに現れる隣接頂点を $\{v_1, \cdots, v_n\}$ とする．各隣接頂点を経由する経路を評価する時点では，辺 (u, v_i) 間のコストは推定コストで評価する．評価関数 $f(u)$ の最小値を与える頂点 $v \in \{v_1, \cdots, v_n\}$ を選択して展開すると，辺 (u, v) の真のコスト $c^*(u, v)$ が観測される．

図 4.3 に示すような部分木 $T_s(v)$ 内の任意の経路 $(v, w, \cdots, u', v', w', \cdots)$ について，前提 4.1 に基づき，構成辺の真のコストは経路に沿って非減少である．したがって，辺 (u, v) と部分木 $T_s(v)$ 内の任意の辺 (u', v') の真のコストについて，推移的に次式が成立する．

$$c^*(u, v) \leq c^*(u', v') \tag{4.8}$$

これは，部分木 $T_s(v)$ 内のすべての辺の推定コスト $c(u', v')$ のうち，$c(u', v') < c^*(u, v) \leq c^*(u', v')$ であるものについては，その推定コストを真値に近づけるために，$c(u', v')$ を $c^*(u, v)$ に置き換える方がよいことを示唆している．そこで，部分木内の各辺の推定コストを次の方法で更新する．k 回目の更新値を $c_k(u', v')$ で表す．

定義 4.6 (辺の推定コストの更新方法)

$$c_{k+1}(u', v') = \max\{c^*(u, v), c_k(u', v')\} \tag{4.9}$$

この更新方法には，次の特徴がある．

補題 4.1

$k = 0, 1, \ldots$ において $c_k(u', v') \leq c_{k+1}(u', v') \leq c^*(u', v')$，すなわち，辺の推定コストは更新により減少することはなく，かつ，その真のコストを超えることもない

証明 (補題 4.1)

まず，更新後の値 $c_{k+1}(u', v')$ は，定義 4.6 により，観測済みの値 $c^*(u, v)$ と更新前の値 $c_k(u', v')$ のいずれか大きい方の値になるため，$k = 0, 1, \ldots$ において $c_k(u', v') \leq c_{k+1}(u', v')$ は自明である．

次に，初期値 $c_0(u', v')$ は，仮定 4.1 により，$c_0(u', v') \leq c^*(u', v')$ であり，かつ，そのように初期値を設定できるから，$k = 0$ において次式が成立する．

$$c_0(u', v') \leq c^*(u', v') \tag{4.10}$$

k 回目の更新値 $c_k(u', v')$ が真のコスト $c^*(u', v')$ を超えないと仮定すると，次式が成立しなければならない．

$$c_k(u', v') \leq c^*(u', v') \tag{4.11}$$

式 (4.9) による $k + 1$ 回目の更新値について，式 (4.8) と式 (4.11) から次式が成立する．

$$c_{k+1}(u', v') = \max\{c^*(u, v), c_k(u', v')\} \leq c^*(u', v') \tag{4.12}$$

式 (4.12) は, $c_k(u', v') \leq c^*(u', v')$ を仮定した場合に, $c_{k+1}(u', v') \leq c^*(u', v')$ が成立することを示している．すなわち，$k = 0, 1, \ldots$ において $c_k(u', v') \leq c_{k+1}(u', v') \leq c^*(u', v')$ である．（証明終）

以後，推定コストを表記する際に，特に必要がない限り更新回数の表記と初期値の区別を省略する．

補題 4.1 から次の補題が示される．

補題 4.2

経路 $(v_i, v_{i+1}, \cdots, v_j)$ の推定コスト $p(v_i, v_j)$，その初期値 $p_0(v_i, v_j)$，および真のコスト $p^*(v_i, v_j)$ について，$p_0(v_i, v_j) \leq p(v_i, v_j) \leq p^*(v_i, v_j)$，すなわち経路の推定コストの値はその初期値以上であり，かつ真のコストの値を超えない

証明 (補題 4.2)

経路 $(v_i, v_{i+1}, \cdots, v_j)$ の各構成辺の推定コスト $c(v_k, v_{k+1})$ は，補題 4.1 から，各辺について $c_0(v_k, v_{k+1}) \leq c(v_k, v_{k+1}) \leq c^*(v_k, v_{k+1})$ である．経路に沿って $c(v_i, v_{i+1})$ から $c(v_{j-1}, v_j)$ まで辺々加え次式を得る．

$$\sum_{k=i}^{j-1} c_0(v_k, v_{k+1}) \leq \sum_{k=i}^{j-1} c(v_k, v_{k+1}) \leq \sum_{k=i}^{j-1} c^*(v_k, v_{k+1}) \tag{4.13}$$

定義 4.1 および定義 4.2 から，$p_0(v_i, v_j) \leq p(v_i, v_j) \leq p^*(v_i, v_j)$ である．　（証明終）

補題 4.3

推定目標コスト $h(v)$，その初期値 $h_0(v)$，および目標コスト $h^*(v)$ について，$h_0(v) \leq h(v) \leq h^*(v)$，すなわち推定目標コストの値はその初期値以上であり，かつ目標コストの値を超えない

証明 (補題 4.3)

頂点 v から到達可能な葉 G_k $(k = 1, \ldots, m)$ への経路 $(v, \cdots, G_k)$ について，補題 4.2 により $p_0(v, G_k) \leq p(v, G_k) \leq p^*(v, G_k)$ である．定義 4.3 から $h^*(v)$ を与える葉が G_{min} であるとすると，$p(v, G_{min}) \leq p^*(v, G_{min}) = h^*(v)$ である．定義 4.4 により $h(v) = \min_{k=1}^{m} \{p(v, G_k)\} \leq p(v, G_{min})$ であるから，次式が成立する．

$$h(v) \leq h^*(v) \tag{4.14}$$

また，$h(v)$ を与える葉が G'_{min} であるとすると，$p_0(v, G'_{min}) \leq p(v, G'_{min})$ だから，定義 4.4 により $h_0(v) = \min_{k=1}^{m} \{p_0(v, G_k)\} \leq p_0(v, G'_{min})$ である．これより次式が成立する．

$$h_0(v) \leq h(v) \tag{4.15}$$

ゆえに，$h_0(v) \leq h(v) \leq h^*(v)$ である．　（証明終）

表 4.1 頂点オブジェクトに関するデータフィールドとメソッド

データフィールド	格納するデータ
$parent$	親頂点のオブジェクト
$fvalue$	この頂点の評価関数値
メソッド	機能
isGoal()	目標頂点であれば True,
	そうでなければ False を返す
SuccSet()	隣接頂点の集合を返す
EdgeSet()	部分木 $T_s(v)$ の辺の集合を返す

表 4.2 リストオブジェクトのメソッド

メソッド	機能
pop()	先頭要素を返す
append(v)	頂点オブジェクト v を要素としてリストに追加する
sort()	要素を評価関数値の昇順にソートする

4.3.4 探索手続

推定コストの更新を含む探索手続の概要を図 4.4 に示す．この探索手続は A*アルゴリズムに推定コストの更新処理を付加したものである．図中の行頭の数字は行番号である．頂点オブジェクトには，表 4.1 に示すデータフィールドとメソッドが定義されているものとする．頂点オブジェクトのリスト構造を構成するリストオブジェクトには，表 4.2 に示すメソッドが定義されているものとする．

S は初期頂点，u は v の親頂点，v は展開対象となる頂点, w は v の隣接頂点にそれぞれ対応する頂点オブジェクトである．$open$ は展開対象となる頂点を評価関数値の昇順に格納するリストオブジェクトである．木構造のすべての辺には，推定コストの初期値が割

```
1  S.parent ← S;
2  g*(S) ← 0;
3  c*(S, S) ← 0;
4  open.append(S);
5  while open ≠ ∅ do
6  |  v ← open.pop();
7  |  u ← v.parent;
8  |  v.fvalue ← g*(u) + c*(u, v) + h(v);
9  |  if v.isGoal() then
10 |  |  return v;
11 |  end
12 |  for (u', v') ∈ v.EdgeSet() do
13 |  |  c(u', v') ← max{c*(u, v), c(u', v')};
14 |  end
15 |  for w ∈ v.SuccSet() do
16 |  |  w.parent ← v;
17 |  |  w.fvalue ← g*(v) + c(v, w) + h(w);
18 |  |  open.append(w);
19 |  end
20 |  open.sort();
21 end
22 return null;
```

図 4.4　提案手法の探索手続き

り当てられているものとする．

まず，初期頂点 S に関する初期化として，S の形式的な親頂点，S から S への真のコストを代入する（行 3）．次に，リスト *open* に初期頂点を追加する（行 4）．以後，*open* から展開する頂点の取り出しと追加を *open* が空になるまで繰り返す．頂点 v を選択するときに観測される $c^*(u, v)$ を用いて評価値を再計算する（行 5-8）．v が目標頂点であるか

どうか終了判定を行う．v の評価値は再計算されているため，v が葉であり，かつ，$open$ の先頭の頂点の評価値が $v.fvalue$ より大きければ，v.isGoal() は True を返し，v を解として探索を終了する．v が葉であり，かつ，$open$ の先頭の頂点の評価値が $v.fvalue$ 以下であれば，v.isGoal() は，v を $open$ に再度追加し False を返す．また，v が葉ではない場合は False を返す（行 9-11）．v が目標頂点でない場合には，$c^*(u,v)$ を利用し，部分木 $T_s(v)$ 内のすべての辺 (u',v') の推定コストを更新する．ただし，$v=S$ のとき，形式的に $c^*(S,S)=0$ としたので，推定コストの実質的な更新はない（行 12-14）．v を展開して得られるすべての隣接頂点 w のそれぞれについて，親頂点を記録したのち，w を経由する経路の評価関数値を算出し，リスト $open$ に加える（行 15-19）．リスト $open$ の要素を評価関数値の昇順に整列する（行 20）．

ここで，推定コストの初期値について言及しておく．真のコストは正の値であるが，推定コストの初期値として 0 を許容するものとする．これは仮定 4.1 を満たすことは明らかである．このとき，探索開始前には，状態空間のすべての推定コストが 0 となる可能性がある．しかし，初期頂点から生成される隣接頂点 $\{v_0,\cdots,v_n\}$ のひとつ v_i が注目頂点として選択された時点で $c^*(S,v_i)$ が観測され，$v_i.fvalue$ の再計算後に目標頂点であるかどうかの検査が行われるため，誤って目標頂点となることはない．また，$c^*(S,v_i)$ の値により，v_i の部分木 $T_s(v_i)$ のすべての辺の推定コストが更新され 0 ではなくなる．$v_i.fvalue=0$ の頂点が $open$ リストにある限り，これが繰り返され，$\{v_0,\cdots,v_n\}$ のすべてが展開された時点ですべての辺の真のコスト，および推定コストは正の値を持ち，かつ，補題 4.1 により，すべての推定コストは仮定 4.1 を満たす．

4.3.5 手法の特性

定理 4.1

提案手法は最適解を発見する

証明 (定理 4.1)

評価関数が $f_{A*}(v)=g_{A*}(v)+h_{A*}(v)$ である A*アルゴリズムを木構造に適用するとき，目標頂点までの推定コストを表す $h_{A*}(v)$ がその真のコストを超えないならば，A*アルゴリズムは最適解を発見して停止することが知られている [5]．提案手法は評価関数が $f(u)=g^*(u)+c(u,v)+h(v)$ であるから，次式が成立すれば A*アルゴリズムと同様に提案手法について最適解の発見を保証できる．

$$c(u,v)+h(v)\leq c^*(u,v)+h^*(v) \tag{4.16}$$

まず，補題 4.1 により，各辺の推定コスト $c(u,v)$ はその真のコスト $c^*(u,v)$ を超えないから，次式が成立する．

$$c(u,v) \leq c^*(u,v) \tag{4.17}$$

次に，補題 4.3 により，推定目標コスト $h(v)$ は目標コスト $h^*(v)$ を超えないから，次式が成立する．

$$h(v) \leq h^*(v) \tag{4.18}$$

式 (4.17) と式 (4.18) の辺々を加え次式が成立する．

$$c(u,v) + h(v) \leq c^*(u,v) + h^*(v) \tag{4.19}$$

これは，式 (4.16) であり，u からどの隣接頂点 v を経由する経路の推定コストも，その真のコストを超えないことを示している．したがって，提案手法は最適解を発見して停止する．（証明終）

定理 4.2

ある状態空間において提案手法が解を発見するまでに展開する頂点数は，同一の状態空間に適用された A*アルゴリズムが解を発見するまでに展開する頂点数以下である

証明 (定理 4.2)

提案手法と A*アルゴリズムのいずれも最適解を発見して停止する．最適解に至る経路の真のコストを f^* とするとき，提案手法では $f(v) \leq f^*$ となる v が Open リストにあれば，v を必ず展開する．A*アルゴリズムもまた，$f_{A*}(v) \leq f^*$ となる v が Open リストにあれば v を必ず展開する．A*アルゴリズムにおける v から目標頂点に至る経路の推定コスト $h_{A*}(v)$ は，更新されないため，提案手法の $h(v)$ 以下である．よって，v が提案手法で展開されるときには，$f_{A*}(v) \leq f(v) \leq f^*$ が成立する．すなわち，提案手法で展開される頂点は，A*アルゴリズムでも展開される．（証明終）

なお，提案手法において，探索中にどの辺の推定コストも初期値から更新されない場合，提案手法の振る舞いは，同一の状態空間に適用される A*アルゴリズムの振る舞いと同じである．そのようになるのは，経路上の隣接辺が (u,v)，(v,w) であるとき，これらの真のコストをそれぞれ $c^*(u,v)$，$c^*(v,w)$，その推定コストの初期値をそれぞれ $c_0(u,v)$，$c_0(v,w)$ とすると，初期頂点から葉に向かうすべての経路の構成辺の間で式 (4.20) の関係が推移的に成立する場合である．

$$c_0(u,v) \leq c^*(u,v) \leq c_0(v,w) \leq c^*(v,w) \tag{4.20}$$

4.4 シミュレーション

4.4.1 実行内容

前提 4.1 を満たす深さ $D \in \{4, 6, 8, 10, 12, 14, 16\}$ の完全二分木を状態空間とする．いずれの深さにおいても，上限を $X \in \{100, 100000\}$ のいずれかとする正の一様乱数 R を用いて，各辺の真のコストの値と推定コストの初期値を付与する．推定コストの初期値 E は, 経路に沿って非減少とは限らない割り当て方法 M_1 と, 真のコストと同様に経路に沿って非減少となる割り当て方法 M_2 の 2 通り $E \in \{M_1, M_2\}$ とする．$(d, x, e) \in D \times X \times E$ である状態空間 $T(d, x, e)$ をそれぞれ 100 例生成する．同一の状態空間に対照手法と提案手法を適用し，それぞれの手法が解を発見するまでに展開する頂点数と，生成する頂点数を記録する．対照手法は A*アルゴリズムであり，推定コストの初期値を使用して経路を評価する．A*アルゴリズムでは推定コストの更新は行われない.

実験 1 は $T(D, 100000, M_1)$，実験 2 は $T(D, 100, M_1)$ を対象として，それぞれ提案手法と対照手法を比較する．また，実験 1 と実験 2 からコストの上限値の差に起因する相違の有無を検討する．実験 3 は $T(D, 100000, M_2)$，実験 4 は $T(D, 100, M_2)$ を対象として，それぞれ提案手法と対照手法を比較する．また，実験 1 と実験 3，実験 2 と実験 4 のそれぞれにおいて，推定コストの初期値の違いに起因する相違の有無を検討する．

4.4.2 状態空間の生成

前提 4.1 が成立する状態空間になるように, 次の方法で各辺にそれぞれの値を付与する.

まず，初期頂点 S とその隣接頂点 $v \in \{v_1, v_2\}$ の間の各辺 (S, v) に，一様乱数 R を用いて真のコストを割り当てる.

次に，再帰的に，既に真のコストが割り当てられた辺 (u, v) について，v とその隣接頂点 $w \in \{w_1, w_2\}$ の間の各辺 (v, w) に対し，一様乱数 R を用いて真のコストを割り当てる．ただし，一様乱数 R で生成される値が，辺 (u, v) に割り当てられている値 r より小さい間は破棄し，最初に現れた r 以上の値を採用する.

すべての辺に真のコストを割り当て終えたら，最後に，すべての辺に推定コストの初期値を割り当てる．実験 1，実験 2 では，各辺の $U = c^*(v, w)$ を上限値として生成した一様乱数の値をその辺 $(v.w)$ の推定コストの初期値として割り当てる．実験 3，実験 4 では，

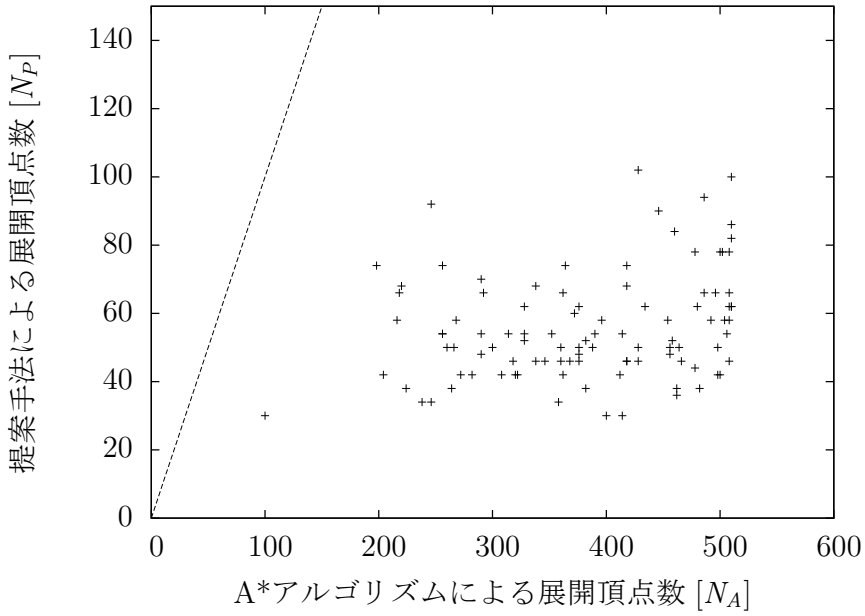


図 4.5　実験 1:展開頂点数の比較（$T(8, 100000, M_1)$）

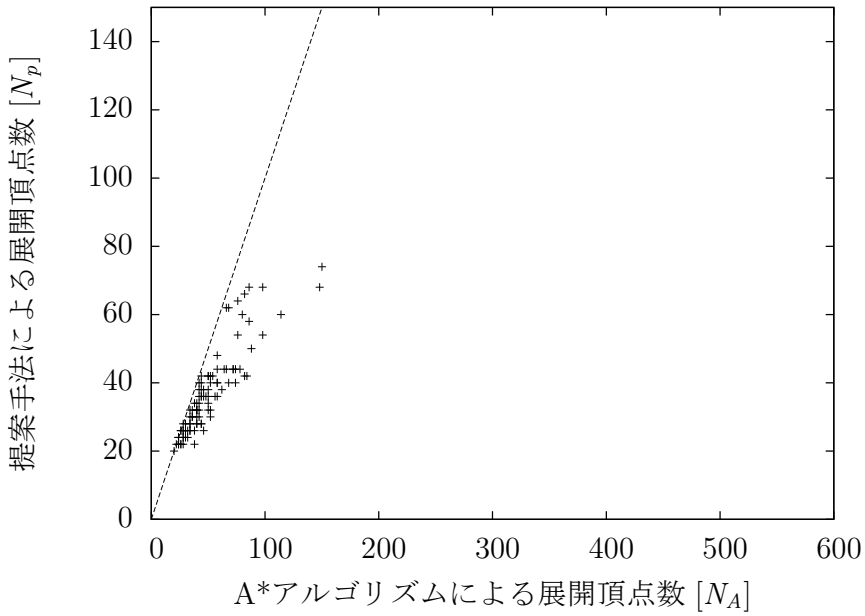


図 4.6　実験 3:展開頂点の比較（$T(8, 100000, M_2)$）

辺 (v, w) の先行辺 (u, v) の推定コスト $L = c(u, v)$ を下限値，真のコスト $U = c^*(v, w)$ を上限値として生成した一様乱数の値を辺 (v, w) の推定コストの初期値ととして割り当てる．

4.4.3 結果と考察

展開頂点数の比較

状態空間は，対照手法による展開頂点数 N_A と，提案手法による展開頂点数 N_P の値の組 (N_A, N_P) により特徴づけられる．横軸に N_A，縦軸に N_P の値をとる座標平面上にプロットすると，プロットされた点がひとつの状態空間 $T(d, x, e)$ に対応する．特に，深さ $D = 8$ の場合について，実験 1 の $T(8, 100000, M_1)$ の結果を図 4.5 に，実験 3 の $T(8, 100000, M_2)$ の結果を図 4.6 にそれぞれ図示する．

いずれの図においても図中の原点を通る直線は，対照手法と提案手法の展開頂点数が等しくなる境界線である．この境界線上とその下側の領域は，提案手法の展開頂点数が対照手法の展開頂点数以下となる領域である．いずれの図においても 100 例すべての状態空間が，境界線上かまたはその下側の領域に存在する．これは定理 4.2 を実験的に示したものである．図 4.6 に見られるように，両手法とも推定コストの初期値が経路に沿って非減少である場合には，それとは限らない場合と比較して，展開頂点数は平均的には減少する．特に対照手法の展開頂点数の減少は顕著であり，提案手法と同数となる場合がある．なお，実験 2 の $T(8, 100, M_1)$，実験 4 の $T(8, 100, M_2)$ についても，それぞれ図 4.5，図 4.6 と同様の傾向を示した．

有効分岐因子の値の比較

深さ $d \in D$ の状態空間ごとに，対照手法と提案手法における 100 例の生成頂点数の平均値 G を用いて，式 (4.21) を満たす有効分岐因子 [29] の値 B をそれぞれ算出し表に示す．

$$B + B^2 + \cdots + B^d = G \tag{4.21}$$

また，横軸に深さ $d \in D$，縦軸に有効分岐因子 B をとり，D と B の関係を図示する．実験 1 の結果を表 4.3 と図 4.7 に，実験 2 の結果を表 4.4 と図 4.8 に，実験 3 の結果を表 4.5 と図 4.9 に，実験 4 の結果を表 4.6 と図 4.10 にそれぞれ示す．

実験 1（表 4.3・図 4.7）は，推定コストの初期値が経路に沿って非減少とは限らない場合の結果である．対照手法は状態空間が深くなるにつれて，有効分岐因子の値がわずかに増加する．これは対照手法では状態空間が深くなるにつれて，解を発見するまでにより広い範囲の探索が必要になることを意味する．状態空間が完全二分木であることを考慮すると，有効分岐因子の上限は 2 であるから，ほとんどの頂点を生成している．一方，提案手

表 4.3　実験 1:生成頂点数と有効分岐因子（$T(D, 100000, M_1)$）

Search tree		A* search		Proposed method	
d	N	G	B	G	B
4	31	25.08	1.89	16.94	1.67
6	127	98.96	1.91	33.94	1.53
8	511	383.30	1.92	55.82	1.43
10	2047	1492.98	1.93	84.76	1.37
12	8191	5951.98	1.94	119.56	1.33
14	32767	23502.12	1.95	161.98	1.30
16	131071	92069.72	1.95	212.44	1.27

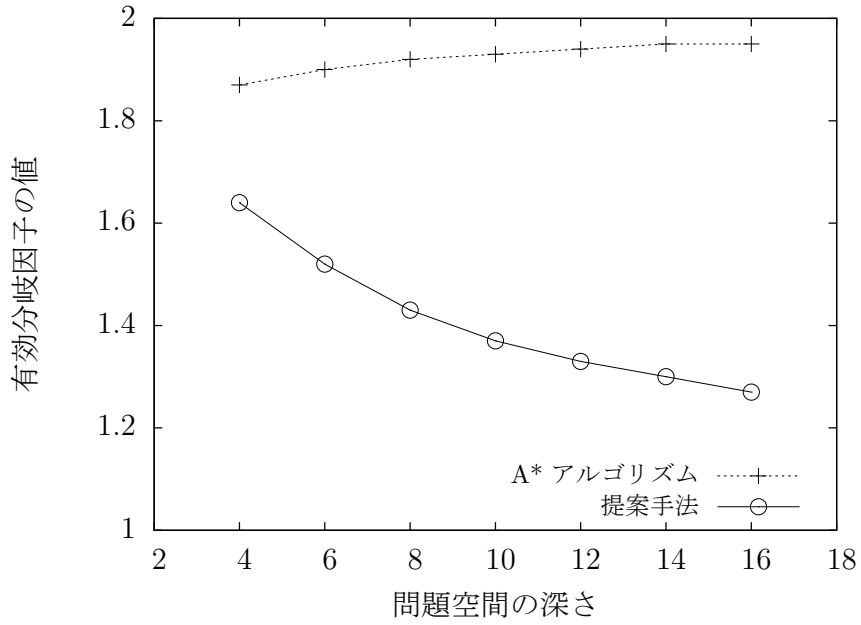


図 4.7　実験 1:有効分岐因子の比較（$T(D, 100000, M_1)$）

法は，状態空間が深くなるにつれて，有効分岐因子の値が減少する．これは状態空間が深くなるにつれてより狭い範囲の探索で解が発見できることを意味する．この傾向は，実験 2 の結果（表 4.4・図 4.8）にも現れる．

実験 3 の結果（表 4.5・図 4.9）は，推定コストの初期値を経路に沿って非減少に設定した場合である．提案手法にはややおよばないものの，対照手法も状態空間が深くなるにつれてより狭い探索範囲で解が発見できることを示している．また，提案手法の有効分岐因

表 4.4 実験 2:生成頂点数と有効分岐因子（$T(D, 100, M_1)$）

Search tree		A* search		Proposed method	
d	N	G	B	G	B
4	31	26.16	1.89	17.04	1.64
6	127	102.30	1.92	32.68	1.51
8	511	394.64	1.91	52.22	1.41
10	2047	1569.42	1.94	76.56	1.35
12	8191	6227.46	1.95	105.02	1.31
14	32767	25106.30	1.96	131.02	1.27
16	131071	97630.80	1.96	165.50	1.25

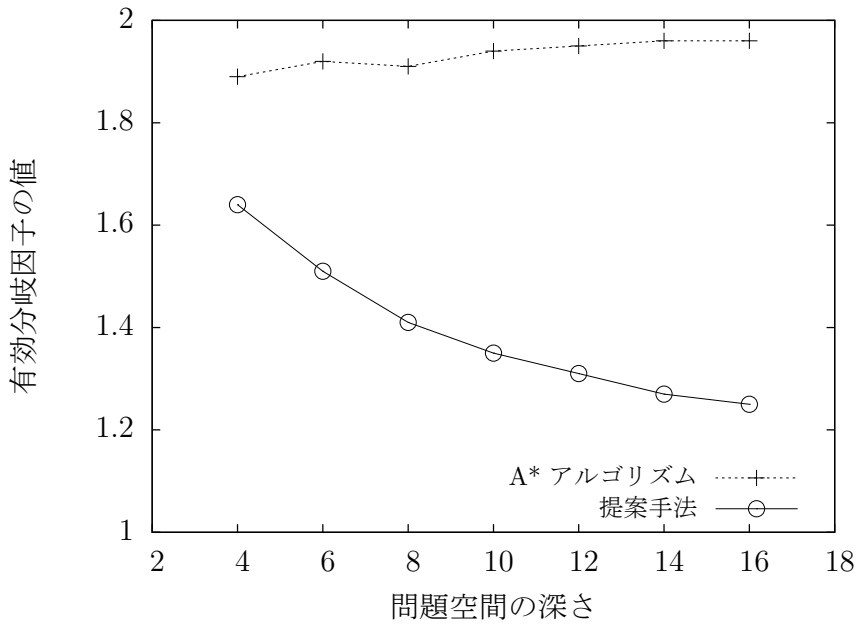


図 4.8 実験 2:有効分岐因子の比較（$T(D, 100, M_1)$）

子の値は，実験 1 の結果よりやや小さい． 推定コストの初期値を経路に沿って非減少となるよう設定すると，それとは限らない場合と比較して，評価関数の精度は高くなる．そのため，対照手法はより狭い探索範囲で解を発見する．提案手法は，推定値の更新により評価関数の精度がさらに高まる．そのため，推定コストの初期値が経路に沿って非減少であるとは限らない場合よりもさらに狭い探索範囲で解を発見する．

実験 4 は，コストの上限値が小さく，かつ推定コストの初期値を経路に沿って非減少に

表 4.5　実験 3:生成頂点数と有効分岐因子（$T(D, 100000, M_2)$）

Search tree		A* search		Proposed method	
d	N	G	B	G	B
4	31	15.24	1.58	13.50	1.51
6	127	30.36	1.48	24.16	1.40
8	511	50.60	1.41	36.74	1.33
10	2047	75.14	1.35	51.16	1.28
12	8191	102.30	1.31	66.58	1.25
14	32767	132.56	1.27	83.40	1.22
16	131071	164.28	1.24	99.50	1.19

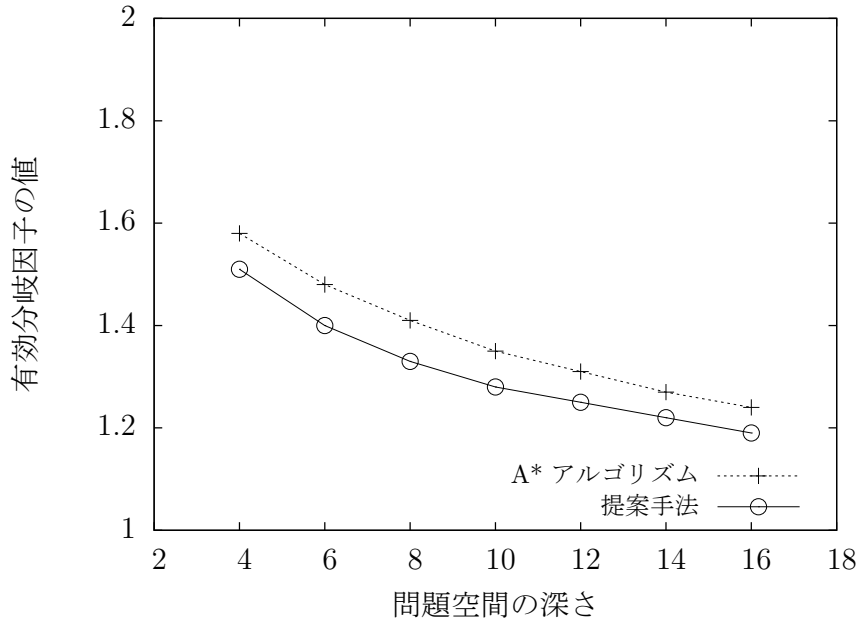


図 4.9　実験 3:有効分岐因子の比較（$T(D, 100000, M_2)$）

設定した場合である．表 4.6・図 4.10 から，対照手法，提案手法ともに有効分岐因子の値が，$d = 10$ を境に増加に転じていることがわかる．頂点 u とその隣接頂点 v が最適経路上にあるとき，次に展開する頂点として v が選ばれると，コストの真の値 $c^*(u, v)$ が観測される．これにより，この時点の最適経路のコストは $f(u)$ から $f(v)$ に $c^*(u, v) - c(u, v)$ だけ増加する．対照手法，提案手法ともに最適解を発見するから，v が展開されるときは，$f(u) < f(w) \leq f(v)$ となるいかなる頂点 w も必ず展開されている．コストの上限値が小

表 4.6 実験 4:生成頂点数と有効分岐因子（$T(D, 100, M_2)$）

Search tree		A* search		Proposed method	
d	N	G	B	G	B
4	31	14.36	1.55	13.18	1.50
6	127	30.22	1.48	24.34	1.41
8	511	50.76	1.41	36.74	1.33
10	2047	80.66	1.36	54.18	1.29
12	8191	157.06	1.37	88.72	1.29
14	32767	399.42	1.40	182.50	1.31
16	131071	1284.48	1.45	510.00	1.36

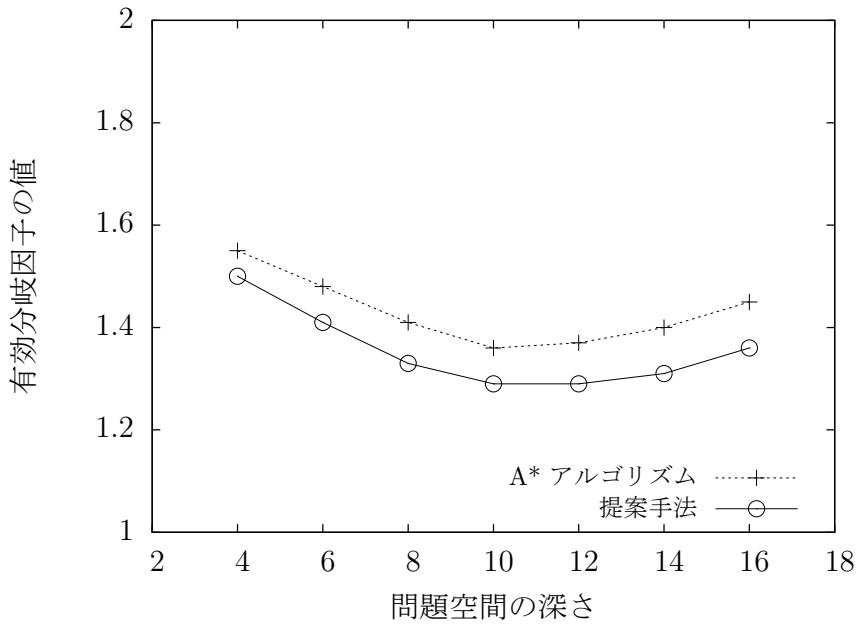


図 4.10 実験 4:有効分岐因子の比較（$T(D, 100, M_2)$）

さいとき，深い状態空間ではコストの真の値として同じ値を割り当てられた辺が多くなり，経路のコストの差が小さくなる．そのため前式を満たす w が多く現れると，展開頂点数とともに生成頂点数が増加し，有効分岐因子の値が増加に転ずるものと考えられる．

4.5 おわりに

葉に向かう経路に沿った辺に順次大きなコストが付与されるという特徴を持った木構造における探索手法を提案した．提案手法は，A*アルゴリズムをベースとして，頂点の選択時に観測される真のコストを利用し，推定コストを更新しながら探索を進めるものである．更新対象になるのは，初期頂点を根とする場合を除き，展開対象となる頂点を根とする部分木内のすべての辺の推定コストである．推定コストは評価関数の適格性を保つように更新するため，最適解の発見が保証される．また，推定コストの更新を伴う評価関数の値は，更新を伴わない場合の値に対し常に支配的である．したがって，提案手法が探索中に展開する頂点数は，同一の状態空間において推定コストを更新しない手法により展開される頂点数以下であることが保証される．これらは A*アルゴリズムの特性であることから，提案手法が A*アルゴリズムを拡張しながらもその特性を継承することを示している．

これらの特性をシミュレーションにより検証した．推定コストを更新しない手法を対照手法として，提案手法と対照手法とを同一の状態空間に適用する．対照手法は推定コストの初期値を評価関数値とする A*アルゴリズムである．状態空間は完全二分木とし，上限値を定めた乱数を利用して，真のコストと推定コストを割り当てる．

まず，提案手法と対照手法において発見される解が一致することにより，各状態空間において最適解を発見していることが確認された．次に，最適解の発見までに展開される頂点数を比較することにより，提案手法による展開頂点数は対照手法のそれ以下であることが確認された．さらに，提案手法の有効分岐因子の値は，対照手法のそれを下回るという優位な結果が得られた．これは，提案手法が最適解を発見するまでの探索範囲が，対照手法のそれより狭いため，効率がよいことを意味する．

真のコストと同様に，推定コストの初期値が経路に沿って非減少である場合，提案手法も対照手法もともに，有効分岐因子の値は，状態空間が深くなるにつれて低下する．推定コストの初期値が経路に沿って非減少とは限らない場合においては，状態空間が深くなるにつれて，提案手法の有効分岐因子の値は低下することに対し，対照手法ではその値は増加する．このことは，本章で対象とする状態空間においては，提案手法は対照手法より推定コストの精度の低さによる影響を受けにくいことを示している．

状態空間を特徴づける一つの要素としてコストの分布を考えると，本章で対象としたような特徴が見られる場合と，そうでない場合とに大別できる．本章では，状態空間全体で

先行する辺よりコストが大きいという特徴を前提としているが，この特徴が一部分だけに現れている状態空間も考えうる．そのような状態空間に対する手法として，提案手法と対照手法を組み合わせ，両手法の長所を活かす手法を考慮することができる．そのためには，状態空間で本章での提案手法が適用可能なコストの特徴が現れているかどうかを判定する仕組みが必要となる．

提案手法は，解を発見するための頂点の展開・生成の処理と，評価関数の更新のための処理から構成されている．対照手法との比較は，おもに前者の処理の結果である展開頂点数と，生成頂点数から見た探査範囲を対象としている．探索の品質という点では，これらは妥当な項目と考えられるが，探索手法の性能という点からの比較では，解の発見までに要する時間や記憶容量を考慮する必要がある．

より現実的な問題への対処という点からは，これらの問題について検討することが課題として残されている．

第 5 章

所与の仕様との逐次的な照合に基づくパターンの獲得

5.1 はじめに

人間は様々な方法を利用して，問題の解決を行うことができる．事前の知識が不十分であり，過去に解決した経験のない，初めて取り扱う問題の場合には，解決に至るまでの指示を受けてから問題解決に着手する方法が考えられる．この方法は実際に作業をしながらその状況を観測し，指示と作業の進行状況との整合性を確認しつつ問題解決を図るものとしてモデル化できよう．本章はこのようにモデル化される問題の解探索について論じるものである．

このモデル化において状態を規定するものは，内・外界センサを用いて観測される作業の進行状況であり，それらは一種のパターンとして獲得されるものとする．見知らぬ土地で現在地から目的地までの道順の指示を頼りに移動することは，この種の問題の典型例であり，経路上の道しるべや地形の特徴などの観測結果が獲得されるパターンとなる．また，道順の指示が目的地に到達するまでに獲得すべきパターンの仕様を与えるものとなる [30][31].

指示が言葉 [32] や実演 [33][34] などで表現される場合には，指示と作業の進行状況との整合性を確認するために獲得したパターンに認識処理を施し（以後，作業認識），指示と照合しうる記述を得る必要がある．作業途中では，その時点までの進行状況を表現する限定されたパターンが観測されるにとどまり，その部分的なパターンには認識対象全体が含まれていない可能性がある．また，認識対象の対象モデルと評価関数が与えられても，一

般にはある程度の認識誤りは不可避である．そのため，作業認識の結果と与えられた指示との整合性の評価は不正確なものにならざるを得ない．これに起因して，与えられた指示に沿って作業が進行していると判断しても，現実には指示どおり進行していないことなどが生じ，問題解決を完了できない恐れがある．

例えば「2 つ目の交差点を右折せよ」という指示が与えられている場合に，図 5.1 の中央付近を隣接する 2 つの交差点とみなすか，1 つの交差点とみなすかで走行経路は異なり（同図，実線および破線の経路），指示された経路に沿って目的地に到達できないことがある．問題解決を完了するためには，指示に沿った作業の実行と進行状況の詳細な観測に加え，過去の作業認識の結果を常に再検討し，状況に応じて過去に遡って作業をやり直す必要がある．

このような指示に基づく問題解決の手法は，部分的なパターンの観測と認識，および指示との照合を逐次的に実行し，指示と整合性の高い全体的なパターンを発見する処理として基本的には定式化できる．具体的には，部分的なパターンの認識結果を頂点とし，作業の進行に伴って成長する木構造において，指示と最もよく一致する認識結果に対応した頂点を探索するプロセスとして実現される．このとき，対象となる状態空間は，探索の進行に伴って構造が変化するものとなる．また，指示との整合性の評価値をコストとみなすと，あらたなパターンの観測によって生じる整合性の評価の変化は，探索中のコストの変化として現れる．

本章ではこのような状態空間を対象とした探索手続きを提案し，その手続きが適切な条件の下で，指示と最もよく一致する認識結果に対応した頂点を必ず発見することを示す．以下第 2 節，第 3 節において，指示に整合するパターンを発見する処理，およびその実現手法を論じ，その特徴に言及する．第 4 節では提案手法を未知環境におけるナビゲーションに適用した動作例を示し若干の検討を加える．第 5 節では，本研究の意義と残された課題について述べる．

5.2　指示に整合するパターンの発見

本章で提案する手法が取り扱う問題の前提条件と，提案手法の概略について述べる．

5.2.1 問題設定

対象となる問題においては，作業の進行状況がセンサを用いて観測可能であり，その観測結果は認識処理を施すことにより，与えられた指示と照合可能な記述が得られることを前提とする．また作業の内容は，適切な手順を経てやり直しが可能であるものとする．

問題解決に至るまでの指示は，あらかじめ定められた記号を用いて記号列に変換する．この記号列は発見すべきパターンの仕様となる．パターンを発見する主体をエージェントと呼ぶ．これは問題解決のための作業を実行する主体に対応する．作業途中ではパターンが部分的に観測される．作業の進行に伴ってパターンの全体が次第に明らかになる．問題解決のための作業手順が異なると，作業の進行状況も一般には異なるため，エージェントが観測可能なパターンは複数存在する．十分に時間をかければ，個々のパターンはその部分的な観測を経て全体が観測可能であり，存在するすべてのパターンも同様にして観測可能であると仮定する．この仮定が成立する作業環境を**有限である**と表現する．

記号列で表現された仕様と照合し整合性を評価するために，パターンには認識処理を施す．認識対象のモデルは付与され，認識結果は記号列で記述する．この記述を**パターンの解釈**と呼ぶ．パターンの解釈と仕様とを照合し，両者の一致の程度（以後，**一致度**）を評価する．一致度の評価関数も与えられるものとする．

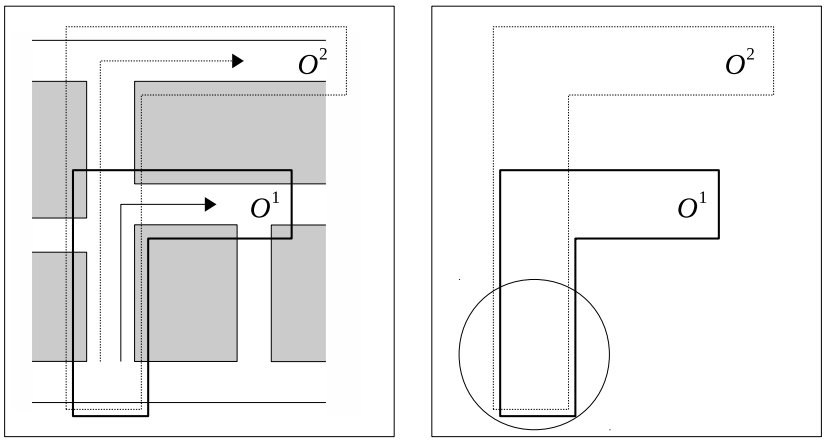
以上の前提の下で，本稿では有限の作業環境に混在する様々なパターンの中から，それらの部分的な観測と認識，および仕様との照合を繰り返し，与えられた評価関数を用いて仕様に最も整合するパターンを発見する探索手法について論じる．

5.2.2 手法の概略

図 5.1 は，指示された経路に沿って環境中を移動する場合に観測されるパターンを，太い実線の囲み O^1 と太い破線の囲み O^2 によって表現した模式図である．円で示された範囲をエージェントの観測範囲とする．

各パターンにおいて観測された部分を**部分パターン** O^i_g，未観測の部分を**残余パターン** $O^i_h(= O^i - O^i_g)$ と呼ぶことにする（図 5.2）．

エージェントの行動に伴ってパターンの新たな部分が観測される（図 5.3）．この部分を**追加パターン** dO^i_h とする（図 5.4（a））．追加パターンと観測済の部分パターンを併合し部分パターンは更新される（$O^i_g \leftarrow O^i_g \cup dO^i_h$）．これに伴い残余パターンも更新される



（a） 環境中のパターン　　（b） パターンと観測範囲

図 5.1　パターンと観測範囲の模式図

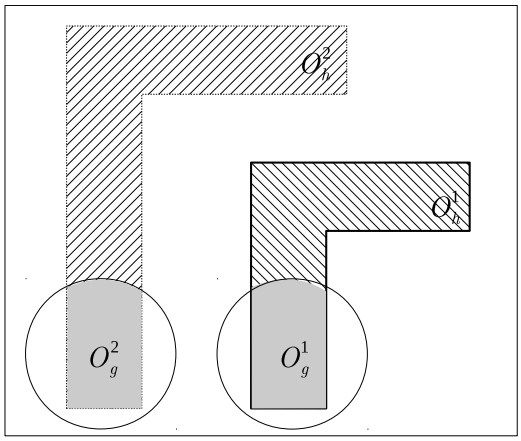


図 5.2　部分パターンと残余パターンの模式図

$(O_h^i \leftarrow O_h^i - O_g^i)$（図 5.4（b））.

パターンの部分的な観測を経て，仕様と整合するパターンの全体を発見するために，本稿では次の手法を提案する．部分パターンに対する認識処理において，その評価の如何に関わらず可能な解釈をすべて生成する．この時点で残余パターンは未観測であるが，実際に観測された場合に得られるであろう解釈を推測する．部分パターンの解釈と残余パターンの推測された解釈の連接をそのパターン全体の解釈とみなす．このようにして得られたすべての解釈と仕様との一致度を部分パターンが更新されるたびに再評価する．最も一致度の高い解釈を生成している部分パターンに対する残余パターンを観測できるような行動を実行し，追加パターンを観測する．以下これを繰り返し，仕様に最も一致する解釈が得

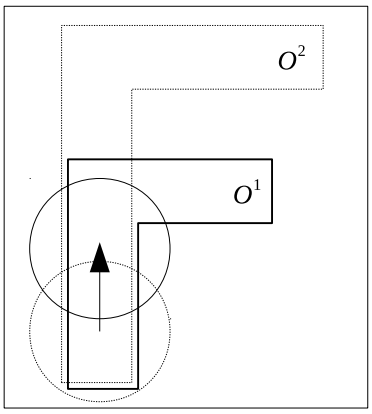
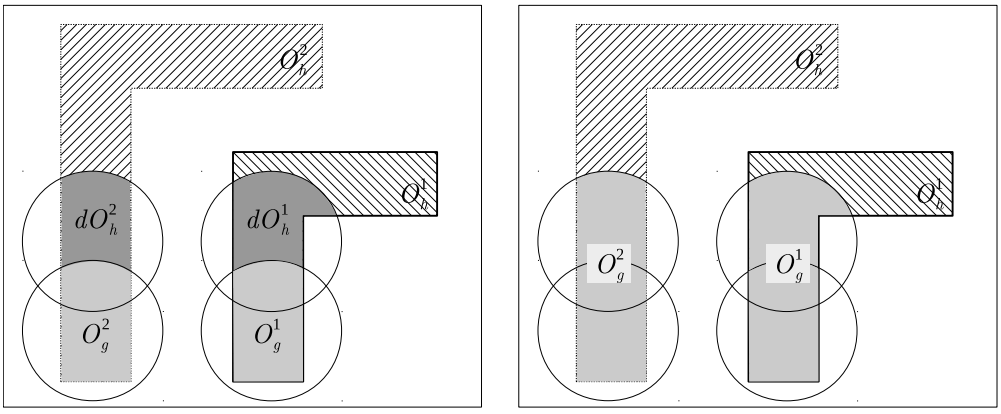


図 5.3 エージェントの行動



（a） 追加パターン （b） パターンの更新

図 5.4 エージェントの行動に伴うパターンの追加と更新

られるパターンを解とみなす．

部分パターンの更新に伴って，ある時点まで仕様と部分的に整合していた部分パターンが整合しなくなることや，あまり整合していない部分パターンが次第に整合しはじめるような変化が生じる．残余パターンから得られる解釈を予測することと，その時点までに生成された解釈と仕様との一致度を常に再評価することによりこの整合性の変化に対応し，対象とするパターンを切替えながら処理を継続することで，仕様と最も整合するパターンの発見を目指す．この一致度の再評価によるパターンの切替えは，過去の認識結果の再検討と，作業の修正・再実行に対応する．

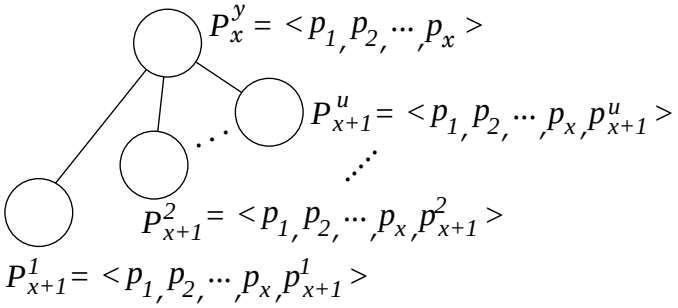


図 5.5　木構造による探索空間の表現

5.3　木探索による手法の実現

前節で述べたように，本章で提案する手法では，状態が過去に遡って再評価される．そこで求解の過程を以下のような木探索として実現する．

5.3.1　解釈木

パターンは複数存在することや，ひとつの部分パターンから複数の解釈が得られることから，パターンの解釈も一般には複数存在すると考えられる．部分パターンに認識処理を施して得られる長さ x の解釈のうち，y 個目のものを記号列 $P_x^y = \langle p_1, p_2, \cdots, p_x \rangle$ と記述する．$p_1, \cdots, p_x$ のそれぞれを**解釈の要素**と呼ぶ．P_x^y と解釈された部分パターンが更新されても，P_x^y に変化は生じない場合と，部分パターンの更新により，P_x^y に新たな解釈の要素 p_{x+1}^i が追加される場合の 2 通りがある．i は追加される要素が幾通りか存在する可能性を示す添字である．新たな要素の追加は，新たな解釈 $P_{x+1}^i = \langle p_1, \cdots, p_x, p_{x+1}^i \rangle$ の生成を意味する．この過程は探索木で表現される．解釈を頂点（状態）とし，部分パターンの更新前の解釈 P_x^y と更新後の解釈 P_{x+1}^i とを辺で接続する．辺が解釈の派生関係を示し，パターンの更新後の解釈に対応する頂点（以下，子頂点と呼ぶ）がパターンの更新前の解釈に対応する頂点（以下，親頂点と呼ぶ）の解釈を継承する木構造となる（図 5.5）．この木構造を解釈木と呼ぶことにする．なお，解釈と頂点は 1 対 1 に対応するため，今後は解釈とそれに対応する頂点の両方を記号 P_x^y で表現する．

部分パターンの更新による新たな解釈の生成に伴い解釈木は成長する．この成長する木構造上で仕様に最も整合する解釈に対応する頂点を探索する．探索の際には，仕様と解釈との一致度を評価する評価関数を定義し，その評価値に基づく探索手法を利用する．

5.3.2 評価関数

仕様と解釈とを照合し，両者の一致度を評価する評価関数を定義する．仕様は記号列 $\langle c_1, c_2, \cdots, c_n \rangle$ で記述する．n は仕様の長さである．仕様を記述する要素 c_i とパターンの解釈 $\langle p_1, p_2, \cdots, p_n \rangle$ の要素 p_i との一致度を評価する評価関数の存在を仮定する．これを**照合関数**と呼ぶ.

仮定 5.1 (照合関数の存在)
仕様を記述する記号 c_i と解釈の要素 p_i を照合し，両者の一致度を $[0, 1]$ の実数値で表現する照合関数 $k_{c_i}(p_i)$ が存在する．一致度が高くなるほど関数値は大きくなるものとする．

仕様と解釈との照合は，先頭の要素から順に実行され，その一致度は次のように定義される．

定義 5.1 (部分パターンの解釈と仕様との一致度)
仕様 $\langle c_1, c_2, \cdots, c_n \rangle$ と部分パターンの長さ x の解釈 $P_x^y = \langle p_1, \cdots, p_x \rangle$（ただし，$x \leq n$）との一致度 $g(P_x^y)$ は照合関数 $k_{c_i}(p_i)$ を用いて次のように表される．

$$g(P_x^y) = \sum_{i=1}^{x} k_{c_i}(p_i) \tag{5.1}$$

仕様の記号列と比較して部分パターンから得られた解釈の記号列が短い場合，仕様の残りの部分と残余パターンから得られるであろう解釈との一致度を推測する評価関数の存在を仮定する．これを**推定値関数**と呼ぶことにする．

仮定 5.2 (推定値関数の存在)
長さ n の仕様 $\langle c_1, \cdots, c_x, c_{x+1}, \cdots, c_n \rangle$ が与えられ，$P_x^y = \langle p_1, \cdots, p_x \rangle$ と解釈される部分パターンが観測されている場合に，その残余パターンから得られるであろう解釈が，仕様の $\langle c_{x+1}, \cdots, c_n \rangle$ の部分に対して，どの程度一致するかを推定する推定値関数 $h(P_x^y)$ が存在する．

照合関数と推定値関数の存在を前提として，部分パターンと残余パターンからなるパターン全体の解釈と仕様との一致度は次のように定義される．

定義 5.2 (パターン全体の解釈と仕様との一致度)
部分パターンの解釈 P_x^y と仕様との一致度 $g(P_x^y)$ と，推定値関数 $h(P_x^y)$ を用いて，パターン全体の解釈と仕様との一致度 $f(P_x^y)$ は次のように表される．

$$f(P_x^y) = g(P_x^y) + h(P_x^y) \tag{5.2}$$

パターン全体の解釈と仕様との一致度の評価関数により算出される評価値に基づいて，仕様と最も整合する解釈に対応する頂点を探索する．

5.3.3 解釈木の探索手法

部分パターンの更新に伴い，その認識結果である解釈と仕様との一致度は変化する．1 つの部分パターンから複数の解釈が得られている場合には，それぞれの解釈と仕様との一致度がすべて変化することや，それぞれの解釈から新たな解釈が派生する可能性がある．ただし，ある解釈から可能なすべての解釈が同時に派生するような部分パターンの更新が行われるとは限らないために，その時点では派生しない場合も起りうる．これらの現象は解釈木上で次の性質となって現れる．

(a)　複数の頂点の評価が変化することがある
(b)　複数の頂点から子頂点が生成されることがある
(c)　子頂点の生成の際に，可能なすべての子頂点が同時に生成されるとは限らない

解釈木上の各頂点は式 (5.2) で評価し，探索手続きはその値が最大となる頂点を探索する．式 (5.2) の形式の評価関数を用いる探索手続きとして，A*アルゴリズムが知られている [8][35]．本稿では解釈木の性質を考慮し，従来の A*アルゴリズムに修正を加えた探索手続きを提案する．

A*アルゴリズムを用いた木構造の探索では，各頂点の評価は初期頂点からその頂点を経由し，目標頂点に至る経路のコストが用いられる．各時点で最も評価の良い頂点のみ展開の対象となる．その頂点から展開可能なすべての子頂点が現れることを前提とし，一旦展開された頂点の評価値は，次にいずれかの頂点が展開されるまで変化することはない．この状況の下で最終的にコスト最小の経路を発見しようとする [36]．一方，本稿で探索の対象となる木構造では，頂点の評価に一致度を用いており，探索は一致度の最も高い頂点を発見するように進められる．頂点の評価はいずれかの子頂点の生成時だけでなく，部分

表 5.1 探索手続きで用いられる副手続き

副手続き	仕様
init(l)	リスト l を初期化する
first(l)	リスト l の先頭要素を取り出す
goal(n)	頂点 n が解であるかどうか判定する．解であれば True を返す．解でなければ False を返す
remove(n,l)	リスト l から頂点 n を削除する
add(n,l)	リスト l に頂点 n を加える
plan(n)	頂点 n の解釈に基づいて観測行動を計画する．計画できる場合はその計画 p を返し，計画できない場合は NULL を返す
action(p)	計画 p を実行し，生成される頂点があれば返す
re-evaluate(l)	リスト l の頂点を再評価する
sort(l)	リスト l の頂点を評価の降順に整列する

表 5.2 探索手続きで用いられる変数

変数名	説明
n	注目頂点
open	次に注目頂点となる頂点の候補のリスト
closed	すべての子頂点が生成された頂点のリスト
p	plan() により生成される計画
v	action() により生成される頂点

パターンの更新に伴って変動する．また頂点の評価とは無関係に，いずれかの頂点で子頂点が生成される可能性がある．さらに子節点の生成時には，一部の子頂点が未生成のまま残されることもある．このような不確定な要素を考慮し，最終的に一致度最大となる頂点を発見するためには，部分パターンを更新する際に必ず頂点を再評価するという修正を加える必要がある．

```
1  init( open );
2    while open ≠ ∅ do
3    | n ← first( open );
4    | if goal(n) then
5    | |  return( n ) ;
6    | end
7    | while plan(n) == NULL do
8    | |  remove( n, open ) ;
9    | |  add( n, closed ) ;
10   | |  n ← first( open );
11   | end
12   | v ← action( plan( n ) ) ;
13   | add( v, open );
14   | re-evaluate( open ) ;
15   | sort( open ) ;
16 end
17 return( False ) ;
```

図 5.6　探索手続き

図 5.6 にこの手続きを示す．表 5.1，表 5.2 はそれぞれ図 5.6 の手続きで使用される副手続きと変数の説明である．init(n) で初期頂点を *open* に追加する（行：2）．*open* リストが空でない間は以降の処理が繰り返される（行：2）．*open* リストの先頭の頂点 n を取り出し（行：3），目標頂点であれば解を発見したものとして終了する（行：4-6）．n が解でない場合には，plan(n) によりその時点で最も評価の高い頂点（以後，**注目頂点**）の評価がさらに高くなる追加パターンの観測を目的とした行動 p を計画する（行：7）．そのような計画ができない場合は，その頂点の子頂点がすべて生成されているものとみなし，その頂点を *closed* リストに移し，次の頂点を取り出すことを繰り返す（行：8-10）．action(p) は計画された行動を実行し，追加パターンを観測した後，部分パターンを更新する（行:12）．その結果，注目頂点であるかどうかに関わらず生成された頂点は *open* リストに追加する（行：13）．*open* リストのすべての頂点を再評価した後（行:14），*open* リスト中の頂点を

評価値の降順に整列させ（行：15）, 注目頂点の選択に戻り，探索を続ける．

なお plan(n) と action(p) は，実際の問題における部分パターンの表現に依存するため，これらの具体的な形式は問題ごとに個別に定義されるものとなる．

5.3.4 探索手続きの特徴

提案する探索手続きは，一致度が最大となる頂点を必ず発見することが示される．仕様の長さを n，長さ n の解釈で仕様と最も一致する解釈（以後，**最適解**）に相当する頂点を P_n^T，長さ n の解釈で P_n^T 以外の解釈（以後，**準最適解**）に相当する任意の頂点を P_n^F，最適解，準最適解以外の任意の頂点を P_x^y とする．頂点 P_n^T の評価値を f^*，頂点 P_x^y に対する推定値関数 $h(P_x^y)$ の真の値を $h^*(P_x^y)$ とする．$h(P_x^y)$ は残余パターンの解釈 $\langle p_{x+1}, \cdots, p_n \rangle$ が存在しない時点での仕様との一致度の推定値である．一方 $h^*(P_x^y)$ の値は，残余パターンが観測され，その解釈 $\langle p_{x+1}, \cdots, p_n \rangle$ が実際に得られた場合の一致度に対応する値である．従って，f^* と $h^*(P_x^y)$ の値は，探索の途中の時点で知ることは現実には不可能である．

まず，図 5.6 の探索手続きにおいて次の補助定理が成立する．

補題 5.1 (最適解の発見)
次の条件が満足されるとき，必ず P_n^T を発見できる．
解釈木のすべての頂点について，

$$h(P_x^y) \geq h^*(P_x^y) \tag{5.3}$$
$$h(P_n^T) = 0 \tag{5.4}$$
$$h(P_n^F) = 0 \tag{5.5}$$

が成立し，かつ，解釈木の根から最適解に至る経路上の頂点 P_x^T が現れる．

証明 (補題 5.1)
P_x^T が存在するにも関わらず，P_n^T を発見せずに終了する可能性がないことを証明する．P_n^F の評価値は式 (5.2) と式 (5.5) から $f(P_n^F) = g(P_n^F) + h(P_n^F) = g(P_n^F)$ である．P_n^F は準最適解であるから，次式が成立する．

$$f^* > f(P_n^F) = g(P_n^F) \tag{5.6}$$

P_x^T の評価値 $f(P_x^T) = g(P_x^T) + h(P_x^T)$ に式 (5.3) を適用すると次式が導かれる．

$$f(P_x^T) \geq g(P_x^T) + h^*(P_x^T) = f^* \tag{5.7}$$

さて，P_x^T が存在するにも関わらず，P_n^F を発見して終了すると仮定すれば，P_x^T と P_n^F の評価値に関して，次式が成立しなければならない．

$$f(P_n^F) = g(P_n^F) > f(P_x^T) \tag{5.8}$$

ここで，式 (5.7) と式 (5.8) から

$$g(P_n^F) > f(P_x^T) \geq f^* \tag{5.9}$$

が導かれる．これは式 (5.6) と矛盾する．ゆえに P_x^T が存在する時に，P_n^F を発見して終了することはない．（証明終）

補題 5.1 は，式 (5.3)・式 (5.4)・式 (5.5) の成立と，最適解に至る経路上の頂点の存在が前提になっているため，これらの成立可能性に言及しておく．

補題 5.2 (最適解・準最適解における推定値関数の値)

頂点 P_n^T と P_n^F に対する推定値関数の値は，$h(P_x^y) \geq h^*(P_x^y)$ に反することなく $h(P_n^T) = 0$，$h(P_n^F) = 0$ とすることができる．

証明 (補題 5.2)

頂点 P_n^T と P_n^F には，仕様 $\langle c_1, \cdots \quad, c_n \rangle$ と照合できる解釈の要素がすべて現れているから，$h^*(P_n^T) = 0$，$h^*(P_n^F) = 0$ である．従って式 (5.3) に反することなく $h(P_n^T) = 0$，$h(P_n^F) = 0$ とすることができる．（証明終）

補題 5.3 (推定値関数の存在)

$h(P_x^y) \geq h^*(P_x^y)$ を満足し，かつ最適解に至る経路上の頂点が探索の過程で必ず現れる推定値関数 $h(P_x^y)$ が少なくともひとつ存在する．

証明 (補題 5.3)

$h^*(P_x^y)$ の値は実際に残余パターンが観測され，$\langle p_{x+1}, \cdots, p_n \rangle$ と解釈された場合の $\langle c_{x+1}, \cdots, c_n \rangle$ との一致度である．仮定 5.1 により，$0 \leq h^*(P_x^y) \leq n - x$ である．ゆえに，次式の推定値関数が存在する．

$$h(P_x^y) = n - x \tag{5.10}$$

さて，最適解に至る経路上の頂点を P_x^T，その子節点を P_{x+1}^y とする．ここで $f(P_x^T)$ と $f(P_{x+1}^y)$ の差 d は次式のようになる．

$$\begin{aligned} d &= f(P_x^T) - f(P_{x+1}^y) \\ &= h(P_x^T) - h(P_{x+1}^y) - \{g(P_{x+1}^y) - g(P_x^T)\} \end{aligned} \tag{5.11}$$

式 (5.10) の推定値関数を用いると，式 (5.11) 第 1 項と第 2 項の部分は

$$\begin{aligned} h(P_x^T) - h(P_{x+1}^y) &= (n-x) - \{n-(x+1)\} \\ &= 1 \end{aligned} \tag{5.12}$$

である．式 (5.11) 第 2 項と，第 3 項の部分は，式 (5.1) と子頂点が親頂点を継承することから次式のようになる．

$$\begin{aligned} g(P_{x+1}^y) - g(P_x^T) &= \sum_{i=1}^{x+1} k_{c_i}(p_i) - \sum_{j=1}^{x} k_{c_j}(p_j) \\ &= k_{c_{x+1}}(p_{x+1}) \end{aligned} \tag{5.13}$$

仮定 5.1 により $0 \leq k_{c_{x+1}}(p_{x+1}) \leq 1$ であるから $d \geq 0$，すなわち

$$f(P_x^T) \geq f(P_{x+1}^y) \tag{5.14}$$

が成立する．このことは，評価値の変動に応じて注目節点が移る場合において，P_x^T の子頂点が注目頂点として選択されるようになるのは，P_x^T の子頂点が全て生成された後であることを示している．P_x^T の全ての子頂点が生成されると最適解に至る経路上の頂点 P_{x+1}^T が必ず含まれる．また，解釈木の根も最適解に至る経路上の頂点のひとつである．従って必ず最適解に至る頂点が現れる．ゆえに式 (5.10) は，探索の過程で最適解に至る経路上の頂点が必ず現れる推定値関数のひとつである． （証明終）

以上から次の定理が成立する．

定理 5.1 (提案手法の最適性)

提案された探索手続きは，仮定 5.1 の条件を満足する照合関数と式 (5.3)・式 (5.4)・式 (5.5)，および式 (5.14) を満足する推定値関数を用いることで必ず解を発見する．

証明 (定理 5.1)

補題 5.1，補題 5.2，および補題 5.3 より明らかである． （証明終）

提案した探索手法が必ず解を発見することは，仕様と最も整合する解釈を与えるパターンの発見を保証することに対応する．そのようなパターンを発見する手法として指示に基づく問題解決の手法が定式化されていることを考慮すると，定理 1 は本稿で対象とする問題に対する問題解決の完了を保証するものと捉えることができる．

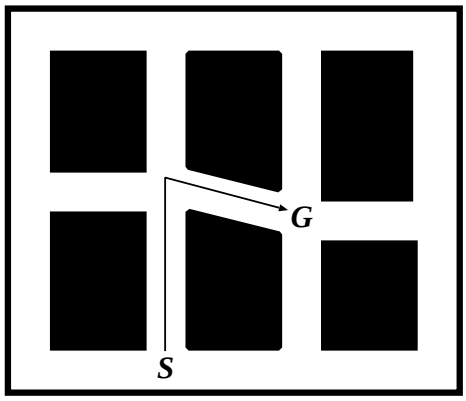


図 5.7　街路状の環境

5.4　未知環境におけるナビゲーションへの応用

本節では，移動経路の指示に基づいて街路状の未知環境を移動するロボットのナビゲーションに提案手法を適用し，その具体的な動作例を示す．

5.4.1　問題設定

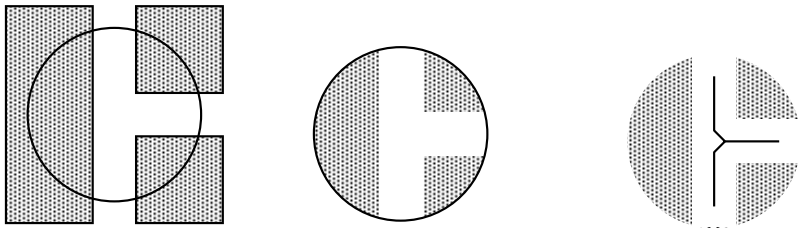
図 5.7 に示す街路状の環境において，S から G への経路を「最初の十字路を右折し，すぐ次の十字路で停止せよ」という指示に従ってロボットを移動させる．移動経路の指示が発見すべきパターンの仕様となり，ロボットの走行経路の観測結果が部分パターンに対応する．仕様と最も整合するパターンを発見することによって，指示された経路に沿った目的地までの移動が達成されることをシミュレーションにより示す．

5.4.2　仕様の記述

移動経路の指示を発見すべきパターンの仕様として次の形式で記述する．

$$\langle (T_1, D_1), (T_2, D_2), \cdots, (T_n, D_n) \rangle \tag{5.15}$$

T_i は経由する交差点の種類，D_i はその交差点での進行方向を示す記号である．種類や進行方向が移動経路の指示に明示されていない個所はいずれも記号 $*$ で表現する．仕様の第 1 項は初期位置とする．なお，この例題での指示「最初の十字路を右折し，すぐ次の十字



（a） 観測範囲　　（b） 部分パターン　　（c） 骨格線と分岐点

図 5.8　部分パターンと骨格線・分岐点の例

路で停止せよ」は次のように記述される．

$$\langle (*,*),(4,\text{右}),(4,*) \rangle \tag{5.16}$$

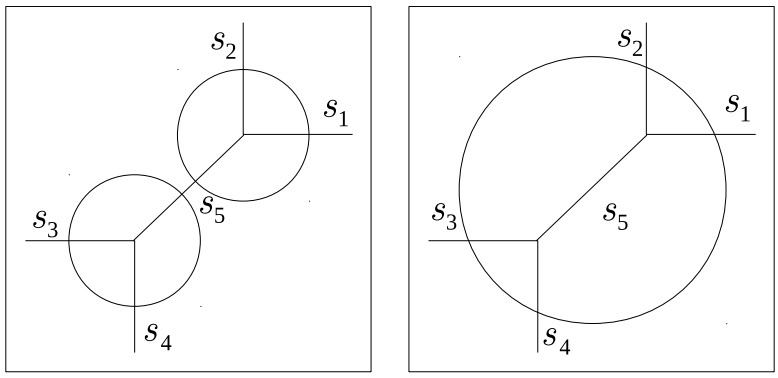
5.4.3　部分パターンと解釈

ロボットは自身を中心とする円内の全方位を観測し（図 5.8（a)），その円内に存在する障害物までの距離を計測する．得られた距離情報に基づき，障害物領域を 1，自由領域を 0 とした占有グリッド表現に変換する [37]．図 5.8（b）は，1 回の観測で獲得される部分パターンの模式図である．画像処理的手法を用いて部分パターンの自由領域の骨格線とその分岐点を抽出する [38] （図 5.8（c)）．抽出された骨格線と分岐点に環境中の通路や交差点を対応させ，部分パターンを部分的な経路として解釈する．

分岐点とそれに接続する骨格線を交差点として認識するとき，複数の認識結果が生じる．例えば図 5.9 の模式図のように 2 つの分岐点が抽出された場合では，図 5.9（a）に示すように分岐点 1 つを 1 つの交差点とする認識結果と，図 5.9（b）に示すように 2 つの分岐点を 1 つの交差点とする認識結果などがある（3 つ以上の分岐点が抽出された場合も同様である)．分岐点が 1 つしか抽出されていない時点でも，それを複数の分岐点からなる交差点の一部分として認識することも可能である．このように分岐点の集合を個々の交差点として認識した結果が解釈の要素であり，それらの列からなる経路全体としての認識結果がパターンの解釈である．

5.4.4　評価関数

1 つの分岐点を 1 つの交差点として認識する場合，その分岐点から延びる骨格線を**外部骨格線**（図 5.9（a）の s_1 から s_5）と呼ぶ．複数の分岐点の集合を一つの交差点として認



（a） 外部骨格線　　（b） 内部骨格線

図 5.9　外部骨格線と内部骨格線の例

識する場合，その集合の要素で隣り合う分岐点間を接続する骨格線を**内部骨格線**（図 5.9（b）の s_5）と呼び，それ以外の骨格線を外部骨格線（図 5.9（b）の s_5 以外）と呼ぶ（今後，両者の区別が不必要である場合は単に骨格線と表記する）．内部骨格線は通路が交差した部分に作られる交差点内部の領域に対応し，外部骨格線は隣接する交差点に至る通路に対応する．

まず，骨格線を通路として認識した場合の評価関数を定義する．骨格線 s_i の長さを l_{s_i}，L を正定数とする．

$$\mathcal{S}(s_i) = 1 - \frac{L}{l_{s_i} + L} \tag{5.17}$$

長い骨格線ほど評価が高くなり，部分パターンの更新に伴って骨格線の長さが変化した場合には，それに応じて評価値が変化する．

次に，1 つ以上の分岐点の集合 V を 1 つの交差点 p_{i_V} として認識した場合の評価関数を定義する．V における内部骨格線の集合を I_V，外部骨格線の集合を E_V，外部骨格線の総数を N_V とする．

$$\mathcal{C}(p_{i_V}) = \left\{ \sum_{s_u \in E_V} \mathcal{S}(s_u) - \sum_{s_v \in I_V} \mathcal{S}(s_v) \right\} / N_V \tag{5.18}$$

ただし分子が負になる場合，評価値は 0 とする．通路に対応する外部骨格線が長く，交差点内部の領域に対応する内部骨格線が短いほど評価は高くなる．交差点を構成している骨格線の長さの変化により，交差点の評価は変化する．

仕様の要素 $c_i = (T_i, D_i)$ との照合対象である交差点 p_{i_V} において，進行方向 D_i に通路

が伸びている場合に 1，それ以外の場合は 0 をとる変数を a_i として，次式のように照合関数を定める．

$$k_{c_i}(p_{i_V}) = \begin{cases} a_i \mathcal{C}(p_{i_V}) & T_i = N_V \\ 0 & T_i \neq N_V \end{cases} \tag{5.19}$$

例えば図 5.8（c）の分岐点の集合 V を 3 差路 p_{i_V} であると認識し，$c_i = (3, \text{右})$ と照合する場合，$k_{c_i}(p_{i_V}) = \mathcal{C}(p_{i_V})$ であるが，$c_i = (3, \text{左})$ と照合する場合は $k_{c_i}(p_{i_V}) = 0$ となる．なお抽出された骨格線の長さは $l_{s_i} > 0$ であるから，式 (5.17) の値域は $(0, 1)$ の実数値である．そのため式 (5.18) の分子は N を越えることはなく，式 (5.19) の照合関数の値域は $[0, 1)$ となり仮定 5.1 の条件を満足する．

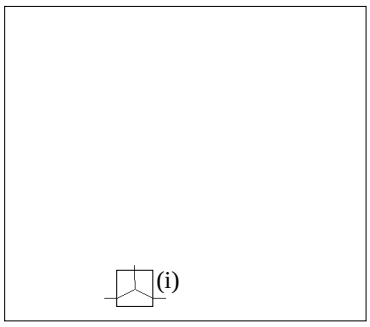
5.4.5　動作例

推定値関数が異なる 2 つの動作例を示す．仕様と最も一致する解釈を発見するエージェントの探索の様子を，解釈木の成長過程の模式図と頂点の評価の変動を記したグラフに示す．また，このエージェントが追加パターンを観測するための行動の様子を骨格線で描かれる地図（以下，走行経路図）の変化として示す．

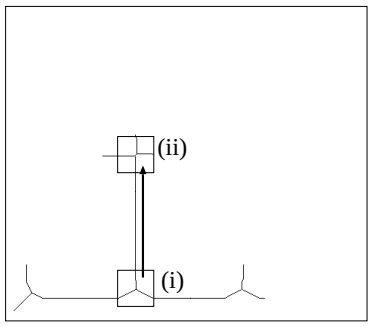
図 5.10，図 5.13 は，エージェントが注目頂点を切り替えた時点と探索を終了した時点の走行経路図である．各図中，実線の矩形の部分は 1 つの交差点として認識した部分であり，破線の矩形は 1 つの交差点の一部分として認識している部分である．これらを太線で接続した部分が，その時点で指示に沿っていると判断した走行経路に対応する．

図 5.11，図 5.14 はエージェントが注目頂点を切り替えた時点と探索を終了した時点の解釈木の模式図である．各時点の注目頂点は灰色で示すが，その他の頂点の多くは模式図への記入を省略した．図中 A,$\cdots$,E の記号が付けられた頂点の解釈は，図 5.10（a）〜 図 5.10（e），および図 5.13（a）〜 図 5.13（c）の走行経路に対応する．

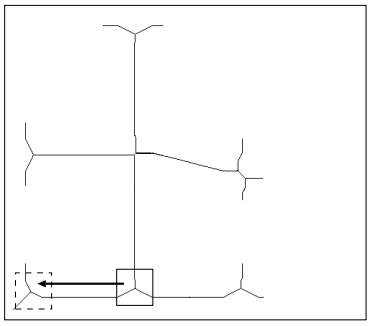
図 5.12（a）, 図 5.12（b），および図 5.15 は横軸に部分パターンの更新回数，縦軸に頂点の評価値をとり，代表的な頂点の評価の変動を示したグラフである．図中 A,$\cdots$,E のラベルがつけられたグラフはそれぞれ図 5.11，図 5.14 の節点に対応する．図 5.12（b）は，図 5.12（a）における頂点 C,D,E のみ表示したグラフである．グラフの開始点は子頂点が生成された時点（解釈が生成された時点）であり，終了点は頂点が *closed* リストに移された時点に対応する．



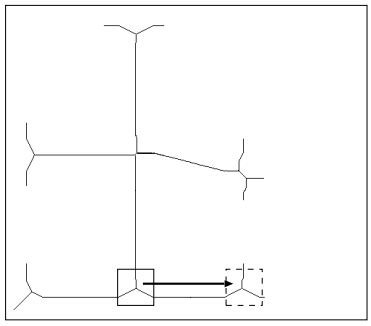
（a）初期位置に対応した解釈を選択



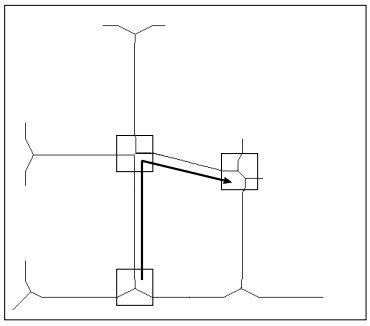
（b）次の十字路までの正しい解釈を選択



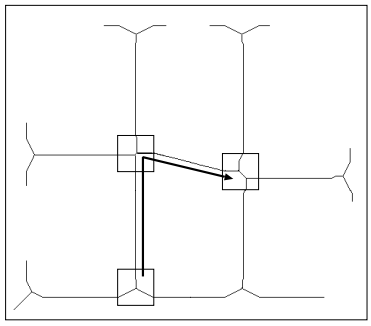
（c）次の十字路までの誤った解釈を選択



（d）次の十字路までの誤った解釈を選択



（e）目的地までの正しい解釈を選択



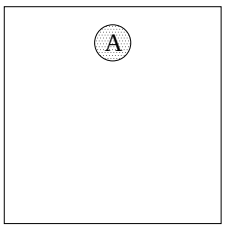
（f）探索終了

図 5.10　動作例 1（走行経路図）

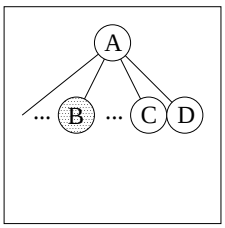
動作例 1

仮定 5.1 を満たす式 (5.19) の照合関数と，式 (5.3)・式 (5.4)・式 (5.5)，および式 (5.14) を満たす式 (5.10) の推定値関数を利用し，定理 1 を満足する評価関数

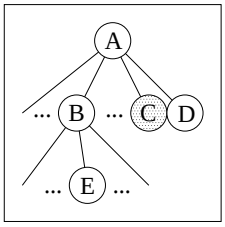
$$f(P_x^y) = \sum_{i=1}^{x} k_{c_i}(p_i) + (n - x) \tag{5.20}$$



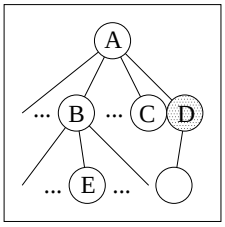
（a）初期位置の解釈に対応する頂点



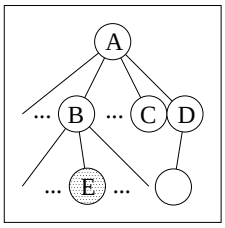
（b）次の十字路までの正しい解釈に対応する頂点



（c）次の十字路までの誤った解釈に対応する頂点



（d）次の十字路までの誤った解釈に対応する頂点

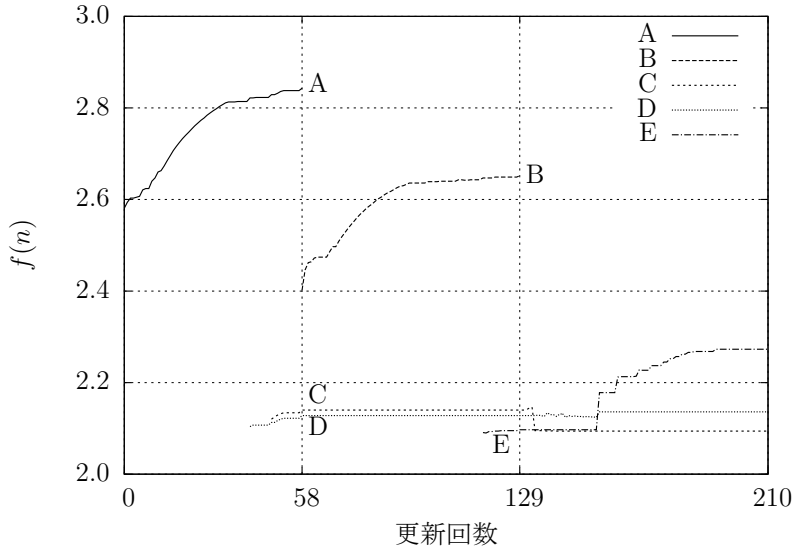


（e）目的地までの正しい解釈に対応する頂点

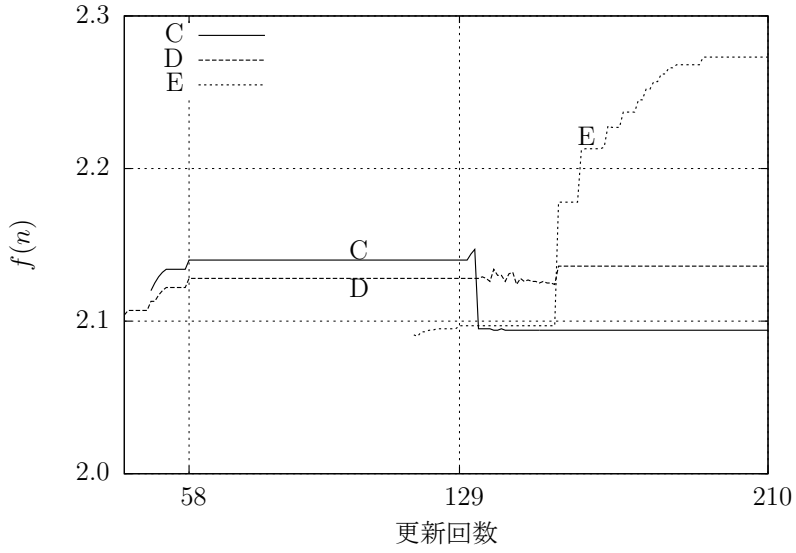
図 5.11　動作例 1（解釈木の摸式図）

により頂点の評価を算出した場合の動作例を示す．

初期位置の交差点を認識した後（図 5.10（a)），まず，同図（i）から右に伸びる骨格線上に分岐点が抽出され，新たな解釈が派生する．次に左に伸びる骨格線上に分岐点が抽出され，新たに解釈が派生する．ここまで注目頂点は変化しない．最後に上に伸びる骨格線上に分岐点が抽出され，さらに新たな解釈が派生し，この時点で注目頂点が頂点 B に移る（図 5.10（b)，ならびに図 5.11（b）の頂点 B．図 5.12（a）の 58 回目の更新時)．探索は継続し図 5.10（b）の（ii）から左，右，上に伸びる骨格線上にこの順序で分岐点が抽出され，それに伴って解釈が順次派生する．その後，注目頂点が頂点 C に移り図 5.10（c）の

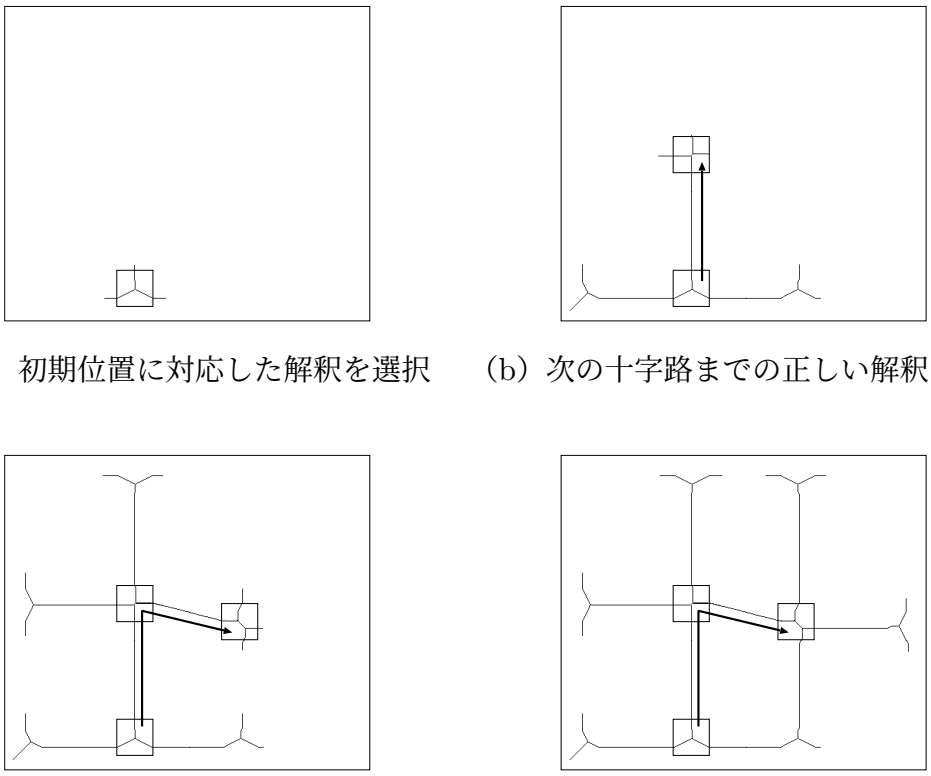


（a）選択された頂点の評価



（b）頂点 C,D,E についての拡大図

図 5.12　動作例 1（頂点の評価）



（a）初期位置に対応した解釈を選択　（b）次の十字路までの正しい解釈を選択

（c）目的地までの正しい解釈を選択　（d）探索終了

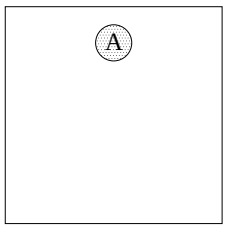
図 5.13　動作例 2（走行経路図）

経路が指示に沿った経路の一部分であると判断する．これは図 5.12（a）のグラフで 129 回目の更新時に対応する．図 5.10（c）の破線の矩形部分付近の追加パターンを観測した後，さらに注目頂点が D 節点に移り図 5.10（d）の経路が指示に沿った経路の一部分であると判断する．図 5.10（d）の破線の矩形部分付近の追加パターンを観測した後，注目頂点は頂点 E に移り図 5.10（e）の経路を指示された経路とみなし，探索を終了する．

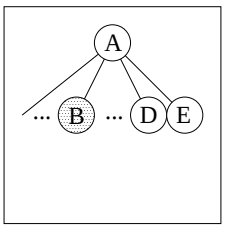
動作例 2

次に，推定値関数の違いによる動作の変化を調べるために，経験的に推定値関数を定めた動作例を示す．

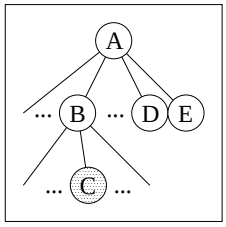
長さ n の仕様に対して，部分パターンから長さ x の解釈が得られている場合，図 5.7 の環境においては，残余パターンの解釈と仕様の残りの部分との一致度は $0.8 \times (n - x)$ 未満となることが実験を通して明らかになった．動作例 2 ではこの事実を反映した推定値関



（a）初期位置の解釈に対応する頂点



（b）次の十字路までの正しい解釈に対応する頂点



（c）目的地までの正しい解釈に対応する頂点

図 5.14　動作例 2（解釈木の摸式図）

数 $0.8 \times (n - x)$ を用いて，次式の評価関数を構成した．

$$f(P_x^y) = \sum_{i=1}^{x} k_i(p_i) + 0.8 \times (n - x) \tag{5.21}$$

図 5.13（b）に至るまでの動作は，動作例 1 の場合と同様である．これ以降，動作例 1 の場合は図 5.10（c），（d）のように指示された経路以外の経路に対応する頂点に注目頂点が移ったことに対し，動作例 2 では図 5.13（c）以降，指示された経路に対応する頂点を注目し続けて探索を終了した．

5.4.6　考察

いずれの動作例においても，最適解を発見して探索を終了した．部分パターンの更新に伴い頂点の評価が変動すること，またその変動に応じて注目頂点が切り替わることが確認された．

2 つの動作例からは，推定値関数により予測される一致度の重要性が示唆される．動作例 1 の推定値関数は，仕様と完全に一致する解釈が残余パターンから得られることを仮定していることに相当する．そのため，実際に残余パターンの一部分が観測され，仕様とあ

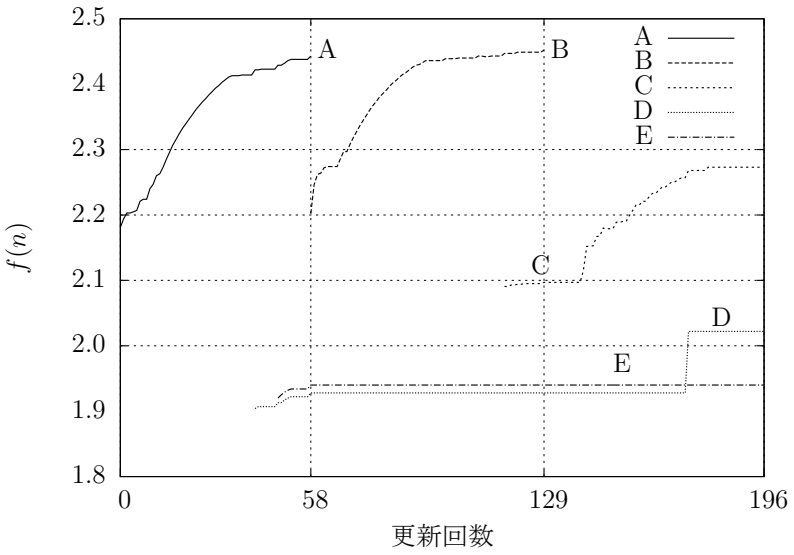


図 5.15 動作例 2（頂点の評価）

まり一致しないことが明らかになるまでどの頂点にも高い評価が与えられる．この結果，最適解に至る経路上の頂点でなくとも注目頂点となる（図 5.11（c），（d））．しかし，動作例 2 の推定値関数のように，残余パターンの解釈と仕様と一致度について，より現実的な評価を仮定すると，最適解に至る頂点を主たる注目頂点とした探索が可能であると考えられる（図 5.14（b），（c））．これは従来の A*アルゴリズムにおいて，評価の推定値が真の評価値に近い程，途中で展開される頂点の数が減少すること [39] に類似する特徴であると推察される．

5.5 おわりに

指示に基づく問題解決の手法が，所与の仕様と整合するパターンを発見する処理として定式化された．具体的な処理はパターンの部分的な観測と認識，仕様との照合を伴う木探索として実現された．提案手法は，

- パターンの実際に観測された部分の解釈と仕様との一致度を評価する照合関数
- 実際の一致度を下回ることなく，パターンの未観測の部分の解釈と仕様との一致度を予測する推定値関数

が存在するという条件の下で，仕様と最も整合するパターンの発見を保証している．このことは作業認識に誤りが生じても，過去に遡って動作を修正し，最終的には問題解決を完了することに対応する．

しかし動作の修正は，必ずしも実際の作業認識の誤りに起因するわけではなく，誤りの可能性を探るために行われることもある．このような動作の修正を減少させるには，作業環境に固有の特徴を反映させるなどの調節を推定値関数に加え，作業の進行状況に対する予測能力を向上させる必要があろう．

本章で議論した木探索の過程は，情報が不十分な状態での一種の思考プロセスのモデル化として捉えることができる．この場合，照合関数は思考過程における中間的な結論の正しさを表現し，推定値関数は経験則の適用の適切さを表現する．これらに基づく頂点の評価の変動は，思考過程で生じる一種の迷いを表現していると考えられる．本章で示した定理が最終的に最適解の発見を保証していることは，思考の収束の保証に対応するものとみなせるであろう．

なお提案手法では，仕様に整合するパターンが複数存在する場合に，仕様の作成者の意図とは異なるパターンを発見して終了する可能性がある．これは，ここでの問題設定で定義された照合関数が，骨格線を対象としてパターンを識別しようとすることによる限界であり，真の一致度の算出が困難であることを示すものとみなせる．この観点からは，一致度をコストと捉えたときに，仕様に整合するパターンが複数存在することを前提とした問題は，照合関数を定義しなおさない限り，探査済みの範囲においてもコストの真値が得られない Class2 の状態空間で表現される問題になる．

従って，類似するパターンの存在が既知である場合には，拘束力の強い仕様すなわち，骨格線以外の要素を用いた認識ができる記述を用いることや，それに合わせて照合関数を定義しなおす必要があると考えている．

本章では，未知環境でのナビゲーションへの応用を試みた．対象とする問題にやや特殊性がみられるものの，この問題の範疇においては，提案手法は比較的一般性の高いものであると考えられる．今後は，ナビゲーション以外の応用可能性を探っていきたい．

第 6 章

結論

6.1 本研究のまとめ

本研究は，コストが付された木構造を対象として，探索品質の向上を意図した評価関数の更新を伴う探索手法の具体例を示し，その特性について論じたものである．

対象とする状態空間は，第 2 章において定めた Class1，すなわち，探査済みの範囲ではコストの真値を獲得でき，未探査の範囲ではコストが推定値にとどまる状態空間である．評価関数は，探査済みの頂点からなる経路のコストを評価するコスト関数と目標頂点に至る未探査の経路のコストを推定するヒューリスティック関数の和で構成する．

コストが付された状態空間における探索では，評価関数を用いて発見した解の品質を評価できる．また，探索中は頂点を網羅的に展開することなく，評価値の高い順に選択的な展開が可能になる．さらに，評価関数の精度を高めることにより，展開頂点数の減少とそれに伴う生成頂点数の減少が見込まれる．

これを踏まえ，探索品質を，探索手法の最適性の有無から判断する解の品質および，探索手法による求解までの展開頂点数，もしくは，生成頂点数から判断する探索過程の品質の 2 点で測るものとし，探索品質の向上を目指した手法を提案した．

第 3 章において，評価関数の精度を高める方法を検討した．ヒューリスティック関数を注目頂点から隣接頂点までの短期予測と，その隣接頂点から目標頂点までの長期予測に分割し，短期予測と長期予測の予測精度をそれぞれ独立に設定する．ここで，最適解と非最適解のそれぞれのヒューリスティック関数の値について，成立することが望ましい大小関係が保たれることを条件とすると，短期予測と長期予測の予測精度が同程度の水準であれ

ば，非最適解と最適解のより正しい識別につながることが導かれた．これは，評価関数の精度を高める上で，コストの推定値と真値との誤差を減少させる方法とは異なる方法があることを示唆する成果であると考えられる．

第 4 章においては，状態空間に付されたコストの特徴を利用し，ヒューリスティック関数を更新する探索手法を提案した．想定した特徴は，辺に付されるコストの真値が，経路に沿って一様に増加するというものである．この特徴のもとでは，注目頂点と隣接頂点を接続する辺のコストの真値が，その隣接頂点を根とする部分木内のすべての辺のコストの真値と同じであるか小さくなる．これを利用し，部分木内の各辺のコストの推定値を適格性が損なわれないように更新する．これにより，コストの真値との誤差を減少させる効果と，長期予測の予測精度を高める効果がある．A*アルゴリズムにこの更新手法を組み合わせた探索手法は，最適性を有することが示された．また，展開頂点数と有効分岐因子の値がともに，対照手法である A*アルゴリズムによるそれらと同じであるか下回ることを実験的に示した．これは，コストに特徴がある状態空間における具体的な探索手法において，ヒューリスティック関数の更新により探索品質の向上が見込まれることを示唆する成果とみなすことができる．

第 5 章では，指示に一致する経路を発見するために，コスト関数の更新を伴う探索手法を提案した．対象とする状態空間は，観測によって獲得した部分的な情報の解釈を頂点とし，その解釈と与えられた指示との一致度をコストとするものである．新たな情報の獲得により，新たな解釈が生成することによって，状態空間に頂点が追加される．また，既存の解釈の一致度が変化することがある．これに対応するために，探査済みの頂点の再探査とコスト関数の更新機能とを A*アルゴリズムに組み合わせた探索手法を提案した．提案手法は，A*アルゴリズムの適格性に類似する条件を満たす場合に，最適解の発見を保証することが示された．これは，状態空間の構造の変化や確定値に至るまでコストの変動が生じる場合でも，頂点の再展開を許容し，コスト関数を更新しながらそれらの変化に対応することにより，最適解が発見できることを示唆する成果とみなせる．

以上から，状態空間の性質に合わせた適切な評価関数の更新は，探索品質の向上に寄与するものといえる．

6.2 今後の課題

ヒューリスティック関数の更新を伴う探索手法においては，状態空間の性質のひとつとしてコストの特徴を考慮した．そこでは，経路に沿ってコストが一様に増加することが前提である．渋滞，遅延，土砂災害の危険度などを対象にした実社会の問題では，対象とすべき領域の一部分については，コストが一様に増加するという特徴が見られるが，全域については，それ以外の特徴も見られる場合があるだろう．このような場合には，本研究の提案手法を別の手法と組み合わせて探索を行うような仕組みが望まれる．そのためには，状態空間においてそれぞれの手法が適用可能となるコストの特徴が発現している部分を判別する仕組みが必要である．

また，提案手法の評価基準として，最適性，展開頂点数，有効分岐因子を用いて，A* アルゴリズムとの比較を行った．最適性を有することと，展開頂点数の優位性が認められることは理論的に保証できている．有効分岐因子の優位性についてはシミュレーション実験で確認したが，推定コストの初期値の与え方に依存して効果に差が出ている．このことと，評価関数の更新に要する計算量を考慮していないことについては，提案手法のより詳細な特性を把握する上でも，解析的な分析を進める必要がある．

参考文献

[1] Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*, pp. 1–12. McGraw-Hill Pub. Co., 1971.

[2] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, pp. 92–102. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[3] Edward F. Moore. The Shortest Path Through a Maze. In *Proceedings of an International Synposium on the Theory of Switching Conference*, pp. 285–292. Harvard University Press, 1959.

[4] Solomon W. Golomb and Leonard D. Baumert. Backtrack Programming. *Journal of ACM*, Vol. 12, No. 4, pp. 516–524, Oct 1965.

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

[6] Joseph C. Culberson and Jonathan Schaeffer. Pattern Databases. *Computational Intelligence*, Vol. 14, No. 3, pp. 318–334, 1998.

[7] Ariel Felner, Richard E. Korf, and Sarit Hanan. Disjoint pattern database heuristics. *Artificial Intelligence*, Vol. 134, pp. 9–22, 2002.

[8] Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*, pp. 75–77. McGraw-Hill Pub. Co., 1971.

[9] Richard E. Korf. Depth-first Iterative-deepening: An Optimal Admissible Tree Search. *Artif. Intell.*, Vol. 27, No. 1, pp. 97–109, Sep 1985.

[10] Edsger W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, Vol. 1, No. 1, pp. 269–271, 1959.

[11] 白井良明. 人工知能の理論, p. 22. コンピュータ数学シリーズ / 斎藤信男, 有澤誠, 筧

捷彦編. コロナ社, 1992.

[12] Rudiger Ebendt and Rolf Drechsler. Weighted A* search - unifying view and application. *Artificial Intelligence*, Vol. 173, No. 14, pp. 1313–1314, 2009.

[13] Maxim Likhachev, Geoffrey J. Gordon, and Sebastian Thrun. ARA* : Anytime A* with Provable Bounds on Sub-Optimality. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pp. 767–774. MIT Press, 2004.

[14] Richard E. Korf. Real-Time Heuristic Search. *Artificial Intelligence*, Vol. 42, No. 2-3, pp. 189 – 211, 1990.

[15] Carlos Hernández and Pedro Meseguer. LRTA*(k). In *Proceedings of the 19th international joint conference on Artificial intelligence*, pp. 1238–1243, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[16] 芦田昌也, 瀧寛和. 道路状況の変動を考慮した走行経路の計画 (「学習」および一般発表). 人工知能学会　知識ベースシステム研究会, Vol. 86, pp. 3–8, Aug 2009.

[17] 芦田昌也, 瀧寛和. コストの変動を考慮した探索手法に関する一考察. 電気学会研究会資料. IIS, 産業システム情報化研究会, Vol. 2009, No. 33, pp. 13–16, Mar 2009.

[18] Gordon J. Vanderbrug. A Geometric Analysis of Heuristic Search. In *Proceedings of the June 7-10, 1976, National Computer Conference and Exposition*, AFIPS '76, pp. 979–986, New York, NY, USA, 1976. ACM.

[19] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*, pp. 32–54. Prentice Hall, Dec 2002.

[20] 安場直史, 長岡諒, 矢野純史, 香川浩司, 森田哲郎, 沼尾正行, 栗原聡. ナビゲーションシステムにおける熟考性と即応性を兼ね備えたルート探索手法. 人工知能学会全国大会論文集, Vol. JSAI08, pp. 334–334, 2008.

[21] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, pp. 103–104. Prentice Hall, 3rd edition, Dec 2009.

[22] 吉岡伸也, 橋本浩良, 上坂克巳. 車線数と交通量が所要時間に及ぼす影響に関する実証的研究. 土木計画学研究・講演集 (CD-ROM), Vol. 39, p. 315, 2009.

[23] 鳥海重喜, 中村幸史, 田口東. 通勤電車の遅延計算モデル. オペレーションズ・リサーチ, Vol. 50, No. 06, pp. 409–416, 6 月 2005.

[24] 仮屋崎圭司, 岩倉成志, 森地茂. 第 93 回運輸政策コロキウム　都市鉄道の運行ダイ

ヤ過密化に伴う列車遅延の波及に関する研究. 運輸政策研究, Vol. 11, No. 4, pp. 111–118, 2009.

[25] 萩原武司, 小澤友記子, 北澤俊彦. データオリエンティッドなイベント時渋滞予測モデル分析. 土木学会年次学術講演会講演概要集 (CD-ROM), Vol. 64, No. Disk2, pp. IV–003, 2009.

[26] 岡田憲治, 牧原康隆, 新保明彦, 永田和彦, 国次雅司, 斉藤清. 土壌雨量指数. 日本気象学会機関誌, Vol. 48, No. 5, pp. 349–356, May 2001.

[27] 芦田昌也, 瀧寛和. 経路の特徴を反映した評価値の更新をともなう木探索. 情報処理学会論文誌, Vol. 54, No. 4, pp. 1632–1640, Apr 2013.

[28] Carlos Hernández and Pedro Meseguer. Improving LRTA*(k). In *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 2312–2317, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[29] Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*, pp. 72–75. McGraw-Hill Pub. Co., 1971.

[30] 芦田昌也, 竹内昭浩, 柴田史久, 角所考, 北橋忠宏. 所与の仕様との逐次的な照合に基づくパターンの獲得. 情報処理学会論文誌, Vol. 40, No. 7, pp. 2872–2881, Jul 1999.

[31] 柴田史久, 芦田昌也, 角所考, 北橋忠宏. 移動ロボットによる言語的経路指示理解のための記号情報とパターン情報の対応付け. 人工知能学会誌, Vol. 13, No. 2, pp. 221–230, Mar 1998.

[32] 岡田豊史, 開一夫, 安西祐一郎. ロボットコマンド学習システム acorn-ii とその評価. 人工知能学会誌, Vol. 9, No. 6, pp. 882–889, Nov 1994.

[33] Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance. *Robotics and Automation, IEEE Transactions on*, Vol. 10, No. 6, pp. 799–822, Dec 1994.

[34] 池内克史, Sing.Bing Kang. 視覚によるハンドの教示. 日本ロボット学会誌, Vol. 13, No. 5, pp. 599–602, Jul 1995.

[35] Stuart.J. Russell and Peter. Norvig. *Artificial Intelligence: A Modern Approach*, pp. 96–101. Prentice Hall International Editions Series. Prentice Hall, 1995.

[36] 白井良明. 人工知能の理論, pp. 19–21. コンピュータ数学シリーズ / 斎藤信男, 有澤誠, 筧捷彦編. コロナ社, 1992.

[37] Jean-Claude Latombe. *Robot Motion Planning: Edition en anglais*. The Springer International Series in Engineering and Computer Science. Springer, 1991.

[38] 長谷川純一, 輿水大和, 中山晶, 横井茂樹. 画像処理の基本技法:技法入門編, pp. 25–69. 技術評論社, 1986.

[39] 上野晴樹, 石塚満. 知識の表現と利用, pp. 143–146. Knowledge representation and use. オーム社, 1987.

謝辞

本論文は，和歌山大学大学院システム工学研究科，ならびに和歌山大学在任中に行った研究をまとめたものです．

和歌山大学大学院システム工学研究科教授 瀧 寛和 先生には，本研究の全般にわたり懇切丁寧なご指導を賜りました．研究の段階ごとに到達すべき目標を明確に示してくださり，進捗状況が思わしくないときでも静かに見守ってくださいました．瀧先生の厚いご指導とご支援，ならびに数々のご助言のおかげをもちまして，本論文を完成させることができました．深く感謝いたします．

本論文をまとめるにあたり，和歌山大学大学院システム工学研究科教授 坂間 千秋 先生には，研究内容を精査いただき，検討が不十分であった箇所をご教示いただくとともに，本研究のさらなる展開に向けたヒントを頂戴いたしました．大阪大学大学院工学研究科教授 馬場口 登 先生 には，研究内容をより深めるためのご指導，およびよりよい論文とするためのご助言をいただきました．和歌山大学システム情報学センター教授 内尾 文隆 先生には，本研究へのご助言をいただくにとどまらず，様々な面でお力添えと励ましをいただきました．深く感謝申し上げます．

筆者の和歌山大学着任後，竹内 昭浩 先生（現，和歌山大学名誉教授）には，研究内容のあり方，学術論文のあり方を説いていただきました．筆者のはじめての投稿論文の執筆過程において，研究内容を探索の問題として定式化することに導いてくださるとともに，論文としてのあるべき姿に少しでも近づくようご指導くださいました．竹内先生のご指導と励ましにより，学位取得の意思を保ち続けることができました．心より感謝申し上げます．

筆者の大阪大学在学中，北橋 忠宏 先生（現，大阪大学名誉教授）には，研究者を目指すきっかけを与えていただくとともに，研究者としての姿勢をご教示いただきました．北橋研究室において知的な刺激を与えてくださったことが，筆者の研究者としての基盤を形成しました．絶えることなくご助言，叱咤激励いただきましたことに心より感謝申し上げます．また，角所 考 氏（現，関西学院大学理工学部教授）には，研究テーマのご提供や研究内容に関するご助言をいただきました．柴田 史久 氏（現，立命館大学情報理工学部

教授）には，研究室での長時間に及ぶ議論のなかで，様々なヒントやアイデアを示してくださいました．厚くお礼申し上げます．

筆者が瀧先生，坂間先生，内尾先生をはじめとして，和歌山大学大学院システム工学研究科の先生方のご指導を仰ぐためにご尽力くださった和歌山大学の教職員の皆様に感謝申し上げます．

最後に，筆者の意思を尊重し支えてくれた両親と姉，いつも明るく迎えてくれる妻と子供に感謝いたします．

本論文の内容に関連して公表した論文など

学会誌掲載論文・ジャーナル論文

1. “経路の特徴を反映した評価値の更新をともなう木探索”,
   芦田昌也，瀧 寛和,
   情報処理学会論文誌，第 54 巻，第 4 号，pp.1632-1640（2013）.
2. “所与の仕様との逐次的な照合に基づくパターンの獲得”,
   芦田昌也，竹内昭浩，柴田史久，角所 考，北橋忠宏,
   情報処理学会論文誌，第 40 巻，第 7 号，pp.2872-2881（1999）.
3. “移動ロボットによる言語的経路指示理解のための記号情報とパターン情報の対応付け”,
   柴田史久，芦田昌也，角所 考，北橋忠宏,
   人工知能学会誌，第 13 巻，第 2 号，pp.221-230（1998）.

国際会議

1. “Geometric Considerations of Search Behavior”,
   Masaya Ashida and Hirokazu Taki,
   KES 2010 Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems : Part II (LNAI6277)，pp.611-619（2010）.

研究会等

1. “動的環境における移動経路の計画に関する一考察”,
   芦田昌也,
   和歌山大学経済学会研究年報 14, 487-495（2010）.
2. “道路状況の変動を考慮した走行経路の計画”,
   芦田昌也，瀧 寛和,
   第 86 回人工知能学会知識ベースシステム研究会，pp.3-8（2009）.
3. “コストの変動を考慮した探索手法に関する一考察”,
   芦田昌也，瀧 寛和,
   電気学会研究会資料, 産業システム情報化研究会, IIS09035, pp.13-16 (2009).