不確実な問題空間におけるリスクを考慮に入れた先読み実時間探索の提案

Real Time Search Taken Risks into Consideration in Uncertain Problem Spaces

*1高橋侑也 Yuya Takahashi　*1*2伊藤孝行 Takayuki Ito

*1名古屋工業大学 Nagoya Institute of Tecnology　*2MIT スローン経営大学院 MIT Sloan School of Management

Real-time search is one of the most effective way when an agent can observe only limited information from its environment. However, if a heuristic value differs from a real value, an agent with these existing algorithms falls into the "wrong" state whose heuristic value is small, and the agent might have difficulty reaching to a goal. In addition, because the existing real-time search algorithms have not considered the dynamic change of problem space, Efficiently solving the search problem in uncertain problem spaces is difficult.In this paper, we propose a real-time search algorithm by looking ahead to avoid falling into a state where the heuristic value is small. In addition, to solve search problems efficiently, we propose a method that determines the action by considering the risks.

1. はじめに

近年，自律移動ロボットの研究が進んでおり，移動の制御を行う際は，空間の状態を把握して移動先を自動的に決定することが必要となる．自律移動ロボットは多くの場合，限られた範囲内の情報しか得ることはできず，その限られた情報をもとに行動を決定する必要がある．以上のような状況で問題を解決するためには，実時間探索アルゴリズムのひとつである Real Time A*(RTA*)[Korf 90] が有効である．

しかし，既存の実時間探索手法において，問題空間の変化について考慮に入れた実時間探索手法はない．既存のアルゴリズムを用いることでも，実時間探索の性質から多少の拡張をすることで不確実な問題空間に適用できるものの，効率良く問題を解くことはできず，またエージェントがとる行動は適切なものであるとはいえない．実世界では時間が経過することで，動作中に環境が変化することが想定される．そのため環境の変化に応じた効率のよい実時間探索手法が必要となる．また，既存の実時間探索手法はヒューリスティック値が小さい地点に陥ってしまい，脱出するのに多くのコストが必要となる．本稿では再帰的に評価値を計算することで，高速かつ容易に先読みを行うことができる手法を提案する．そしてエージェントが問題空間の変化する確率を認識できる状況において，効率の良く問題を解けるように，リスクを考慮に入れ評価値に重みを付けることで，不確実な問題空間においてより適切に問題を解けるように拡張する．

2. 関連研究

経路探索手法はオフライン探索と実時間（オンライン）探索の二種類に分類される．深さ優先探索や A*などのあらかじめ経路をすべて計算するオフライン探索とは異なり，実時間探索は探索と動作を交互に繰り返すことで解に到達する手法である．オフライン探索は A*や最小コスト優先探索を用いることで最適解を得ることができるが，すべての状態を把握し，現在までの経路を記憶しておく必要があり，多くのメモリが必要となる．一方，実時間探索は最適解を保証できないものの，少量のメモリで探索可能であり大規模な問題空間にも適用可能である．またエージェントが問題空間のすべてを把握できない状況にも適用可能であり自律移動ロボットの経路探索などに応用可能である．またターゲットが移動する状況にも対応であり MTS[Ishida 92][Ishida 91]．以下に代表的な実時間探索アルゴリズムである RTA*のアルゴリズムを示す。現在のノードを x，隣接するノードを x'，x と x' 間のコストを $d(x,x')$ とする．

(1) すべての隣接するノード x' について評価値を計算する．ただし $h(x)$ は x のヒューリスティック値，$d(x,x')$ は x，x' 間の距離とする．

$$f(x') = h(x') + d(x,x')$$

(2) ヒューリスティック値 $h(x)$ を $f(x')$ の二番目に小さい値に更新する．

(3) 最小の $f(x')$ をあたえる x' に移動する．最小の $f(x')$ をあたえる x' が複数ある場合はランダムに選択する．

以上 (1)〜(3) を繰り返すことでゴールまで到達する．RTA*は移動コストが正であればヒューリスティック値 $h(x)$ は単調増加するため，いずれかは必ず誤った経路から抜け出すことができる．そのため，RTA*は正の移動コストと有限な推定評価関数と問題空間において完全性が証明されている．RTA*を繰り返し実行することで最適解を求める Learning Real Time A*(LRTA*)[石田亨 96] や，ターゲットの移動を考慮に入れた Moving Target Search(MTS) などの手法もある．

オフライン探索の分野では問題の変化を考慮に入れた Lifelong Planning A*[Koenig 04] や D*Lite[Koenig 08] などが提案されている．これらの手法は問題空間の変化を考慮に入れているものの，変化をした際には，ゴールまでの経路を再計算しており，頻繁に問題が変化する状況では，非常に大きなコストがかかるという問題がある．そのため本研究では，実時間探索において問題の変化を考慮に入れた手法を提案する．

3. 不確実な問題空間の定義

本論文では不確実な問題空間をネットワーク型の問題空間として扱う．ネットワーク型の問題空間は状態を表すノードと，

連絡先: 高橋侑也，名古屋工業大学産業戦略工学専攻，takahashi@itolab.mta.nitech.ac.jp

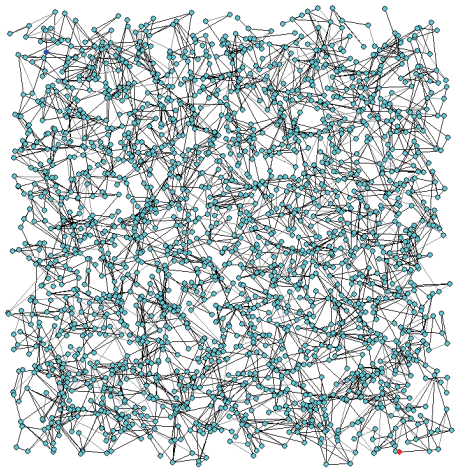

図 1: 1000 ノードの問題空間

ノード間のつながりを表すエッジで構成される．ネットワーク型の問題空間は汎用性が高く，さまざまな問題に適用できると考えられる．またノードを現在の状態，エッジをエージェントの行為とみなすことでプランニングの分野にも適用できると考えられる．ネットワーク型の問題空間は図 1 のようになる．ネットワーク型の問題空間においては，ノードの数が同じであれば，エッジの数が少ないほど目標状態へと到達可能な経路の数が少なく問題を解くことが難しくなり，逆にエッジの数が多いほど目標状態へと到達可能な経路が多くなり問題を解くことが簡単になる．

変化する問題空間としては，エッジの状態が有効，無効と変化する状況を想定する．それぞれのエッジについて破壊確率 (エッジの状態が有効から無効に変化する確率 $R_{destroy}$) と再生確率 (エッジの状態が無効から有効に変化する確率 R_{repair}) を設定する．以上の確率のもとで 1 ステップごとにエッジの状態が変化する状況を想定する．エッジの状態の遷移は図 2 のようになる．本論文では破壊確率，および再生確率は一定であり，時間によって変化しないものとする．

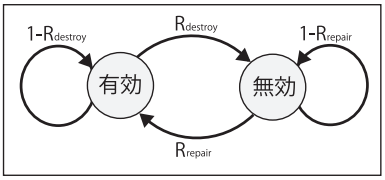

図 2: エッジの状態の遷移図

エージェントはエッジの破壊確率，再生確率が分かるものとし，それらの情報を用いて探索を行う．またエージェントは隣接するノードについて，エージェントの現在のノードとそれらのノードに接続するエッジの状態が有効であっても，無効であっても隣接するノードの状態を認識できるものとする．

4. 提案手法

本研究では不確実な問題空間において，問題を効率的に解くために，再帰を用いた先読みとリスクを考慮に入れた評価値の重み付けの手法を提案する．

4.1 再帰を用いた先読み

まず，誤った経路に侵入しやすいという問題を解決するために，再帰を用いた先読みについて述べる．ここではエージェントが深さ n まで認識できる状況を想定する，そしてエージェントは n 以下である d まで先読みを行い探索を行う．深さが d 未満である深さ i のノードについて，探索には深さ $i+1$ の評価値を用いる．評価値は以下のように決定する．ただし，$parent(x)$ はノード x の親ノード，$child(x)$ はノード x の子ノードとする．

(1) 深さが d である場合

$$f(x) = h(x) + d(parent(x), x)$$

(2) 深さが d 未満である場合

$$f(x) = min_{x' \in child(x) \cap x' \notin parent(x)}\{f(child(x))\} + d(parent(x), x)$$

評価値を計算する際には親ノードを除外している．これは親ノードを除外するのは閉路を回避するため，循環路を回避するため，また探索するノードを少なくすることで探索時間を縮小できるようするためである．探索木を作成して通常のオフライン探索を用いて先読みを行う手法もあるが，再帰を用いることで，非常にシンプルになり，実装が容易である．またオフライン探索を用いる場合は最小となるノードを求めるために，経路を記憶しておく必要があるが，再帰を用いれば経路を記憶しておく必要はないためオフライン探索を用いるより有利であると考えられる．

4.2 リスクを考慮に入れた重み付け

次に不確実な問題空間において効率的に問題を解くために，リスクを考慮に入れた手法を提案する．リスクの高い経路については，評価値に重みを付けることで，効率的に問題を解くことができると考えられる．

本研究で定義した問題空間においては破壊確率が大きいほど移動不可能となる確率が高くなるためにリスクが高くなり，また再生確率が大きいほど移動可能となる確率が高くなりリスクは少なく，また移動不可能である場合でも，少ない時間で移動可能となるためリスクは少なくなる．

本研究ではエッジの破壊確率，および再生確率を利用し，待ち時間の期待値を計算することでリスクを評価し評価値に重みを付ける．以下にノード x における待ち時間の期待値の計算方法について述べる．もしノード x へと移動可能であれば待ち時間は 0 となる．ノード x へと移動不可能である場合は待ち時間の期待値は以下のように計算できる．

$$\sum_{k=0}^{\infty} k * (1 - R_{repair}(x))^{k-1} * R_{repair}(x) = \frac{1}{R_{repair}(x)}$$

また探索深さが i であるノードの未来の状態を推測するために $i-1$ ステップ後のノード x へと移動不可能となる確率を計算する．この確率は現在の状態とエッジの変化確率から計算できる．この確率を $p_{invalid}(x,t)$ とする．以上から，待ち時間の期待値は以下のように計算できる．

$$p_{invalid}(x) * \frac{1}{R_{repair}(x)}$$

となる．この値を評価値の計算時に用いる．評価値計算は，待ち時間の期待値に関数を掛け合わせることで行う．関数はノー

ドの次数を引数とした値とする．これはノードの次数が大きいほどゴールに至る経路の数が多く，ノードの次数が小さいほどゴールに至る経路が少ないと考えたためである．待ち時間を考慮に入れた評価値 $fw(x)$ は以下のように計算する，ただし $g(x)$ は x の次数を引数とする関数である．

$$fw(x) = f(x) + g(x) * p_{invalid}(x) * \frac{1}{R_{repair}(x)}$$

エージェントは $fw(x)$ の値が最も小さいノードを次の移動先として決定する．もしも最小の重み付き評価値を与えるノードに移動不可能である場合は待機する．

4.3 アルゴリズム

図 3 に上記の 2 つの手法を組み込んだ先読み RTA*アルゴリズムを示す．

以下にアルゴリズム内の各関数の働きについて述べる．**Main()** は最初に一度だけ呼び出される関数である．ゴールに到達するまで探索，動作を繰り返す．**MoveOneStep()** はエージェントの一連の行動を行う関数である．ヒューリスティック値の更新を行い．隣接するノードの評価値を計算し，最小の評価値を与えるノードを調べる．そして，そのノードへと移動可能な場合は移動し，移動不可能である場合は待機する．**Update(x,d)** はノード x のヒューリスティック値の更新を行う関数である．戻り値は更新後のヒューリスティック値である．**CalcEval(x,d,parent)** はノード x の評価値を計算する関数である．d は現在の探索の深さであり，$parent$ はノードのリストである．戻り値は計算された評価値である．**CalcWait(x,d)** はノード x における待ち時間の期待値を計算する関数である．戻り値は計算された待ち時間の期待値である．

5. 評価と考察

本研究では 1000 ノードのランダムで作成した問題空間を用いて実験を行った．ノードの破壊確率および再生確率を 0.1 から 0.5 までの一様分布としている．探索深さを変化せせることで再帰を用いた先読みの評価を行う．また評価値のリスクに対する重みを変化させることで，リスクに対する重み付けがどの程度有効に働くのかを評価する．

5.1 再帰を用いた先読みの実験・評価

探索深さを 1 から 7 まで変化させることで実験を行った．$g(x) = 10$ とする．結果を図 4 に示す．横軸は 1 ノードあたりに接続するエッジ数を表す，この数が少ないほどゴールへと至る経路が少なく問題を解くことが難しくなる．縦軸は累計ステップ数を表し，この数が少ないほど効率よく問題を解けていることを表す．

探索深さ 1 は先読みを行っていない方法である．先読みを行うことでかなりの効率化が図れることが分かる．1 ノードあたりのエッジの接続数がどのような値であっても，探索深さが深いほうが，探索深さが浅い場合よりも常に効率よく問題を解くことができる．先読みによる効率化が最も顕著なのは探索深さを 1 から 2 に増加させた場合で，探索深さを 2 から 3，4 から 5 と増加させていくとだんだんと変化の幅が小さくなっていくことが分かる．本実験では探索深さ 5 から探索深さ 7 までに，それほど大きな変化は見られない．探索深さを深くしていくと評価値計算時間が指数的に増加していくため，計算時間も増加していく．したがって，先読み実時間探索では探索深さを深くすることで効率化が図れるが，計算時間についても考慮に入れる必要がある．

```
1. function Main()
2.     x ← start
3.     while agent searches the goal
4.         MoveOneStep();

5. function MoveOneStep()
6.     Update(x,0);
7.     z ← arg min_{x'∈neighbor(x)} CalcEval(x',0,{x})
8.     if(validity(z) == true)
9.         x ← z

10.function Update(x,d)
11.    if(x == goal)
12.        return(0);
13.    else if(d ==(n-1))
14.        h(x) = min_{x'∈neighbor(x)}(h(x') + d(x,x'));
15.        return(h(x));
16.    else
17.        h(x) = min_{x'∈neighbor(x)}(Update(x',d+1)+d(x,x'));
18.        return(h(x));

19.function CalcEval(x,d,parent)
20.    parent.add(x);
21.    if((x' ∈ neighbor(x) ∩ x' ∉ parent) == φ)
22.        f = ∞;
23.    else if((d ==(n-1)) or (x == goal))
24.        f = min_{x'∈neighbor(x)∩x'∉parent}
               [h(x')+d(x,x')+g(x')*CalcWait(x',d)];
25.    else
26.        f = min_{x'∈neighbor(x)∩x'∉parent}
               [CalcEval(x',d+1,parent)+d(x,x')
                                  +g(x')*CalcWait(x',d)];
27.    return(f);

28.function CalcWait(x,d)
29.    if(validity(x)==true)
30.        valid ← 1; invalid ← 0;
31.    else
32.        valid ← 0; invalid ← 1;
33.    for(count=1;count < d;count++)
34.        newValid ← (1 − R_destroy(x)) * valid
                                  + R_repair(x) * invalid;
35.        valid ← newValid;
36.        invalid ← 1 − newValid;
37.    return(invalid/R_repair(x));
```

図 3: 先読み RTA*アルゴリズム

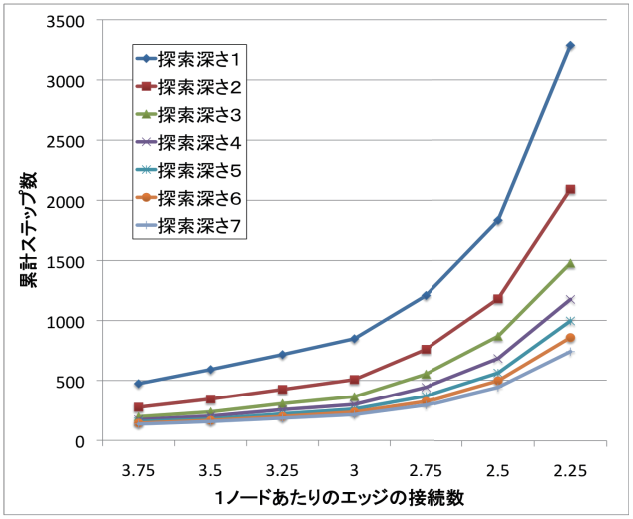

図 4: 先読み性能の評価

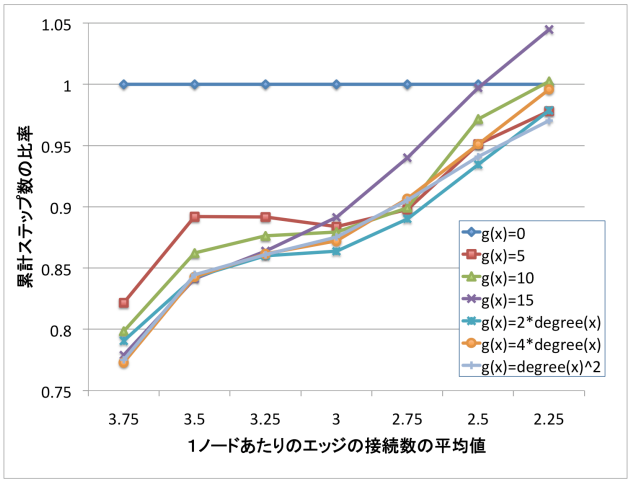

図 5: リスクを考慮に入れた評価値重み付けの評価

5.2 リスクを考慮に入れた評価値重み付けの評価・実験

探索深さを２とし，評価値の重みを付ける際の $g(x)$ の値を変化させることで実験を行った．$g(x)$ の値は定数である 0,5,10,15，そしてノードの次数を引数とした関数である $2 * degree(x)$，$4 * degree(x)$，そして $(degree(x))^2$ の計７種類について実験を行った．

結果を図５に示す．横軸は１ノードあたりに接続するエッジ数を表す，縦軸は g(x)=0，つまり問題の変化を考慮に入れていないものを基準とした累計ステップ数の比率を表し，この数が少ないほど効率よく問題を解けていることを表す．

１ノードあたりのエッジ数によって実験結果に大きな差が出た．全体的に見ると１ノードあたりのエッジの接続数が少ないときには大きな効果が出て，エッジの接続数が少なくなるにつれて効果が薄くなってきている．これは１ノードあたりのエッジの接続数が小さくなると，おおよその場合が待機するのが最良となり変化を考慮に入れる余地が少なくなっているためであると考えられる．$g(x)$ の値が整数であるとき，１ノードあたりのエッジ数が多く問題が簡単である場合には $g(x)$ の値が大きい方が効率的であり，逆に１ノードあたりのエッジ数が少なく問題が複雑である場合は $g(x)$ の値は小さい方が効率的である．

次に $g(x)$ の値がノードの次数を引数とした関数により与えられるときを考える．ここでも１ノードあたりのエッジ数が多く問題が簡単である場合には $g(x)$ の値が大きい方が効率的であり，逆に１ノードあたりのエッジ数が少なく問題が複雑である場合は $g(x)$ の値は小さい方が効率的である．

$g(x)$ が整数である場合と，ノードの次数を引数とした関数である場合を比較すると，どちらも似たような動きをしているが，ノードの次数を引数とした関数を利用している方が，全体的に効率良く解けていることが分かる．これは重みにノードの次数を利用することで，ノードの次数が大きいときには待ちにくく，ノードの次数が小さいときには待ちやすく調整することで，より柔軟に行動を決定できるためであると考えられる．

6. まとめ

本研究では不確実な問題空間において効率よく問題を解くことができるように，再帰を用いた先読み手法，そしてリスクを考慮に入れた評価値の重み付け手法を提案した．またシミュレーション実験を行い，先読みが正しく行われ，高速化ができていること，またリスクを考慮に入れ，評価値計算の際に待ち時間の重みを付けることで，効率的に問題を解けることを示した．

本研究で扱った変化する問題空間は，変化する問題空間の一つであるため，他の状況についても考慮に入れた手法を提案すること，また既存の研究ではゴールが不確実な状況については想定されていないため，問題空間が不確実である状況だけでなく，ゴールが不確実である状況についても考慮に入れることも今後の課題である．

参考文献

[Ishida 91] Ishida, T. and Korf, R. E.: Moving target Search, *IJCAI* (1991)

[Ishida 92] Ishida, T. and Korf, R. E.: Moving-Target Search with Intelligence, *AAAI*, pp. 525–532 (1992)

[Koenig 04] Koenig, S., Likhachev, M., and Furcy, D.: Lifelong Planning A*, *Artificial Intelligence* (2004)

[Koenig 08] Koenig, S. and Likhachev, M.: D* Lite, *AAAI* (2008)

[Korf 90] Korf, R. E.: Real-Time Heuristic Search, *Artificial Intelligence*, Vol. 42, No. 2-3, pp. 189–211 (1990)

[石田亨 96] 石田亨 新保仁：実時間探索による経路学習, *Journal of Japanese Society for Artificial Intelligence*, Vol. 11, No. 3, pp. 411–419 (1996)