# 不確実な問題空間における先読みに基づく実時間探索手法

高橋 侑也 †
†名古屋工業大学情報工学科

伊藤 孝行 †‡
‡MIT スローン経営大学院

## 1 はじめに

既存の研究では，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でも多少の拡張を行う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．実世界では時間の変化などにより，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．本稿では先読みを行うことで効率的に問題を解くことを可能とする先読みRTA*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，待ち時間を考慮に入れた実時間探索手法を提案する．

## 2 関連研究

RTA*[1] に代表される実時間探索ではプランニングと動作を交互に実行する．A*などのオフライン探索とは異なり最適解を保証できないが，少量のメモリで実行可能で，問題空間や目標状態が変化するときにも適用可能である．以下に代表的な実時間探索アルゴリズムであるRTA*を示す．

1）現在のノード $x$ に隣接するすべての $x'$ について，ヒューリスティック値 $h$ とコスト $d$ を用い評価値 $f$ を計算する．

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

2）ヒューリスティック値を $f(x')$ の二番目に小さい値に更新する．

$$h(x) = second\_min\{f(x')\}$$

3）最小の $f(x')$ を与える $x'$ に移動する．

また変化する問題空間についてはプランニングの分野から Lifelong Planning A*(LPA*)[2] が提案されている．LPA*はあらかじめ最短経路を探索し，問題が変化した場合に再探索を行う手法である．ゴールまでの経路を探索しているため，一度の探索に必要なコストは大きい．

†Yuya TAKAHASHI ‡Takayuki ITO
†Dept. of Computer Science, Nagoya Institute of Technology
‡MIT Sloan School of Management

## 3 不確実な問題空間における先読み RTA*

本研究ではネットワーク型の問題空間を想定し，エッジが有効，無効と切り替わる状況を想定する．それぞれのエッジについて，再生確率，および破壊確率を設定し，エージェントはそれらの確率が分かるものとする．本研究ではこれらの確率は一定であるものとする．またエージェントが $n$ 個先のノードまで認識できる状況を想定する．

先読みRTA*のアルゴリズムを図1に示す．$neighbor(x)$ は $x$ の隣接するノードを表し，$varidity(x)$ は $x$ へと移動可能か，不可能かを表す．また $r_r(x)$，$r_d(x)$ をそれぞれ $x$ の再生確率，および破壊確率とする．

```
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')+α*CalcWait(x',d)];
25.    else
26.       f = min_{x'∈neighbor(x)∩x'∉parent}
                 [CalcEval(x',d+1,parent)+d(x, x')+α*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_d(x)) * valid + r_r(x) * invalid;
35.       valid ← newValid;    invalid ← 1 − newValid;
36.    return(invalid/r_r(x));
```

図 1: 先読み RTA*アルゴリズム

**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．

ヒューリスティック値の更新，および評価値の計算は再帰を用いて計算している．深さが $n$ 未満であり，またゴールでない場合はノードを展開し再帰的に評価値を計算する．再帰を用いることで，シンプルな構造で問題を解くことが可能となり実装も容易である．評価値計算の際には隣接するノードからそのノードに至るまでの経路に含まれるノードを除いている (Line 22,24)．これにより行き止まりや循環経路を除外できるため，効率的に探索をすることが可能となり，また不要なノードの展開を避けることができるため評価値計算を高速化できると考えられる．

**CalcWait(x,d)** により計算された値を評価値計算の際に $\alpha$ の重みで重み付けをしている．$\alpha$ は定数であり，$\alpha$ の値が大きいほど移動不可能となりやすいノードの評価値を大きく見積もり，$\alpha$ の値が小さいほど移動不可能となりやすいノードの評価値を小さく見積もる．

ノード $x$ における待ち時間の期待値は以下のように計算できる．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．

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

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 (Line29 35)．したがって待ち時間の期待値は計算された $i-1$ ステップ後のノード $x$ へと移動不可能となる確率を $p_{invalid}(x)$ とすると，

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

となる．この値を評価値の計算時に用いる (Line 36)．

## 4 評価と考察

1000 ノードのランダムで作成した問題空間において実験を行った．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，待ち時間重み定数 $\alpha$ を 10 に設定する．探索深さを 1 から 7 までとし，それぞれの深さについて探索の性能を評価する．

結果を図 2 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．

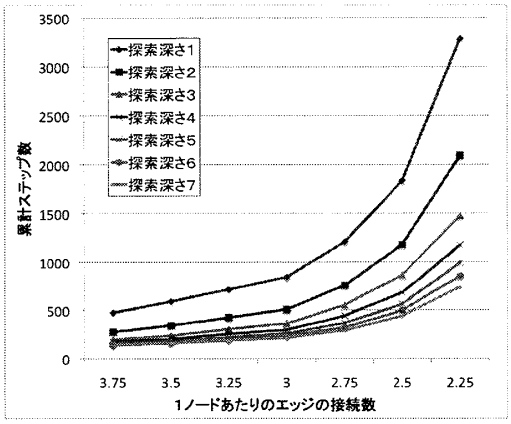


図 2: 実験結果

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，探索深さが浅い状況と比較して常に効率よく問題を解くことができる．深さ 7 で探索を行った場合，先読みを行わない方法と比較すると，4 倍程度の効率化ができる．

## 5 まとめ

本研究では先読みを行うことで，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先読み RTA* を提案した．また待ち時間を考慮に入れることでさらに効率的に問題を解けるように拡張を行った．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は今後の課題である．

## 参考文献

[1] Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, Vol. 42, No. 2-3, pp. 189–211, 1990.

[2] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong planning a*. *Artificial Intelligence*, 2004.