多状態実時間両方向探索

Multi-State Real-Time Bidirectional Search

越野 亮

Makoto Koshino

石川工業高等専門学校電子情報工学科

Department of Electronics and Information, Ishikawa National College of Technology
koshino@isikawa-nct.ac.jp

村田 裕章

Hiroaki Murata

(同 上)

白山 政敏

Masatoshi Shirayama

(同 上)

shira@ishikawa-nct.ac.jp

木村 春彦

Haruhiko Kimura

金沢大学工学部情報システム工学科

Dept. of Information and System Engineering, Faculty of Engineering, Kanazawa University
kimura@ec.t.kanazawa-u.ac.jp

keywords: heuristic search, bidirectional search, real-time A*, commitment

Summary

This paper presents Multi-State Real-Time Bidirectional Search (MSRTBS), a method that improves the efficiency of the heuristic search algorithm for finding approximate solutions. Real-Time A* (RTA*) is a representative heuristic search algorithm for finding approximate solutions. The Multi-State Commitment (MSC) method was introduced into RTA* and dramatically improved the performance in problems such as the N-puzzle. As well, Real-Time Bidirectional Search (RTBS) also improved RTA* by changing a unidirectional search into a bidirectional one. This paper proposed a method that introduces MSC into RTBS. The experimental results showed that compared with RTBS and MSC our proposed method, MSRTBS, improved executed time and solution quality in the N-Puzzle.

1. は じ め に

人工知能における探索とは、状態空間グラフにおける初期状態から目標状態までの経路（解）を求めることである。診断・設計・計画など様々な分野へ応用され、最も基礎的な問題であるが、探索の計算複雑度は NP-完全または NP-困難であるため、状態空間のサイズが大きくなると現実的な時間で解を求めることが困難になる。

問題に関するヒューリスティックな知識を評価関数として用いることで効率的な探索を実現した手法として、A*アルゴリズム [Hart 68] がある。A*アルゴリズムは最適解を発見することを保証するが、膨大なメモリ量と計算時間が必要になる。実用的な観点から考えると、時間をかけて最適解を求めることよりも、許容可能な時間内に近似解を得ることが望ましい場合が多い。そのような近似解探索アルゴリズムとして、RTA*（実時間探索：Real-Time A*）[Korf 90] がある。RTA*は、先読み探索と移動を交互に行うアルゴリズムであり、先読みにより探索範囲を限定することで、計算量は改善される。しかし、次に展開する状態を一つに限定するため、選択を誤った場合の悪影響は大きい。

誤った選択に対する頑健性を向上させる方法として、

複数の問題解決器（エージェント）が RTA*アルゴリズムを並行して実行するマルチエージェント実時間探索 (MRTA*: Multiagent RTA*) [北村 96] と呼ばれる手法が提案されている。また、文献 [横尾 97] では、MRTA*において、遺伝的アルゴリズムの淘汰に似た方法を用いてエージェントを再配置することにより、N パズルにおいて劇的な性能改善が可能であることを示している。しかし、MRTA*では複数のエージェントが同一の状態を展開し、冗長な探索を行う可能性が高くなるという問題点がある。多状態コミットメント探索 [北村 99] では、RTA*や WA* [Pohl 70] などの単一エージェントによる探索で n 個の状態を保持することで、MRTA*のそのような問題点を解消している。

一方、初期状態と目標状態から同時に実時間探索を実行する実時間両方向探索 (RTBS: Real-Time Bidirectional Search) [Ishida 96] がある。RTBS では、初期状態と目標状態から出発した 2 つの問題解決器（エージェント）が、常に他方の問題解決器を目指して移動する。RTBS は単方向実時間探索と比べ、15 パズルや 24 パズルにおいて、性能が改善されている。

本論文では、実時間両方向探索に多状態コミットメントを導入した多状態実時間両方向探索 (MSRTBS: Multi-

State Real-Time Bidirectional Search) を提案する。また, N パズルと迷路問題における性能評価実験を行い, 有効性を検証する。実験結果では, N パズルにおいては, MSRTBS は従来の多状態コミットメント探索や実時間両方向探索と比べ, 大幅に性能を改善できたが, 迷路問題においては, MSRTBS においても同様に有効ではなかったが, 従来の RTBS と比べると性能は改善されることがわかった。

2. 従 来 手 法

本章では, まず, ヒューリスティック探索の定式化を行い, 従来手法である多状態コミットメント探索, 実時間両方向探索について簡単に述べる。

2.1 ヒューリスティック探索の定式化

【定義 1】問題を四つ組 $\langle S, O, s_0, G \rangle$ で定義する。ここで,

S : 状態の集合

O : 状態遷移関数の集合

s_0 : 初期状態 ($s_0 \in S$)

G : 目標状態 ($G \in S$)

である。

組 $\langle S, O \rangle$ で定義されるグラフを状態空間グラフと呼ぶ。ある状態 $s (\in S)$ にオペレータ $o = (s, s') (\in O)$ を適用して得られる状態 s' を s の子状態といい, それを得ることを生成するという。また, 状態 s の全ての子状態を得ることを状態 s を展開するという。

オペレータの適用にコスト (非負の整数) が付いており, 状態 s, s' 間の遷移のコストを $c(s, s')$ で表す。オペレータの連続的な適用により得られる状態の系列は経路と呼ばれ, 初期状態から目標状態への経路を解 (経路) と呼ぶ。解を構成するオペレータのコストの和を解のコストとする。ある解が存在するとき, それよりも小さいコストをもつ解が存在しなければ, その解を最適解と呼ぶ。

ヒューリスティック探索では, 各状態 s から目標状態 G までのコストの推測値が与えられており, ヒューリスティック関数 $h(s)$ で表される。

2.2 多状態コミットメント探索 (MSC-RTA*)

RTA*[Korf 90] は先読み探索と移動を交互に行う近似解探索アルゴリズムである。RTA*は次に展開する状態を一つに限定することで高速に探索することができるが, 選択を誤った場合の悪影響は大きいという問題があった。そこで, 次に展開する状態を複数個 (n 個) 保持することで, 選択を誤った場合の影響を軽減する手法として, 多状態コミットメント探索 (MSC-RTA*: Multi State Commitment - RTA*)[北村 99] がある。

RTA*アルゴリズムには, オンライン探索とオフライン探索の 2 つの実装方法がある。オンライン探索は, 定数時

間の探索結果を基に判断し, 移動しながら探索を続けるアルゴリズムであり, オフライン探索は目標に至る経路を予め完全に探索するアルゴリズムである。MSC-RTA*では, 目標状態に到達した後, 求まった解経路から冗長な経路を取り除き, 最適解に近い近似解を求めるオフライン探索を用いている。

MSC-RTA*では RTA*と同様に展開した状態を保存する *CLOSED* リストは用いず, 生成された状態を保持する *OPEN* リストのみを用いる。そのため, 何度も同じ状態を生成することになるが, 同じ状態に到達するたびにその状態の推定距離の値を単調増加させるようにすることで, いずれループから脱出できるようにしている。以下に, MSC-RTA*のアルゴリズムを示す*¹。

< MSC-RTA*アルゴリズム >

Step1: 初期状態 s_0 を *OPEN* に加える。保持する状態の数 n を設定する。

Step2: if (*OPEN*=空) then 探索失敗, 終了。

Step3: *OPEN* の先頭の状態 s を取り出す。

Step4: if (s =目標状態 G) then 探索成功, 終了。

Step5: s を展開し, 子状態 s' を全て生成する。

Step6: 全ての s' に関して $h(s')$ を計算。

Step7: 全ての s' を *OPEN* に加え, *OPEN* を昇順にソートする。

Step8: if (*OPEN* の個数 $> n$) then n 個になるまで, *OPEN* の中で最大の評価値の状態を削除する。

Step9: $h(s)$ の値を $c(s, s') + h(s')$ のうち 2 番目に小さい値に更新する。ただし, s' の数が 1 個ならばその状態を再展開する意味がないので, に更新する。

Step10: Step2 へ。

2.3 実時間両方向探索 (RTBS)

RTBS[Ishida 96] は, 初期状態と目標状態の両方向から同時に双方の状態に向かって実時間探索を実行するアルゴリズムである。

RTBSでは, プランニングと行動を交互に実行するエージェントのための手法であるため, オンライン探索を用いているが, 本研究では, 効率的に高精度の近似解を求めることを目的としているため, オフライン探索で実装する。

両方向の現在位置を (x, y) , 両方向間の推定値を $h(x, y)$ とする。計算用のリストを $TEMP_1, TEMP_2$ とする。以下に RTBS のアルゴリズムを示す。

< RTBS アルゴリズム >

Step1: 初期状態 s_0 を x , 目標状態 G を y とする。

Step2: if ($x = y$) then 探索成功, 終了。

Step3: if ($TEMP_1 = \text{空}$) then

(1) x を展開し生成される全ての子状態の集合を $TEMP_1$ に加える。

*1 MSC-RTA*は $n = 1$ のとき, RTA*と等しくなる。

- (2) すべての $x' \in TEMP_1$ に関して $h(x', y)$ を計算する .

Step4: if ($TEMP_2 = \text{空}$) then

- (1) y を展開し生成される全ての子状態の集合を $TEMP_2$ に加える .
 (2) すべての $y' \in TEMP_2$ に関して $h(x, y')$ を計算する .

Step5: 以下の式に従い, $h(x, y)$ を更新する .

$$h(x, y) = \text{secondmin}_{x', y'} \begin{cases} c(x, x') + h(x', y) \\ c(y, y') + h(x, y') \end{cases}$$

($h(x, y)$ の値を $c(x, x') + h(x', y)$ と $c(y, y') + h(x, y')$ のうち 2 番目に小さい値に更新する .)

Step6: 前向き探索と後向き探索の制御戦略

- (1) if ($\min\{h(x', y)\} < \min\{h(x, y')\}$) then 最小の $h(x', y)$ を持つ x' を x とし, $TEMP_1$ を空にする (最小のものが複数個あればランダムに選ぶ)
 (2) if ($\min\{h(x', y)\} > \min\{h(x, y')\}$) then 最小の $h(x, y')$ を持つ y' を y とし, $TEMP_2$ を空にする (最小のものが複数個あればランダムに選ぶ)
 (3) if ($\min\{h(x', y)\} = \min\{h(x, y')\}$) then ランダムで (1)(2) を実行する .

Step7: Step2 へ .

3. 提案手法 : 多状態実時間両方向探索

本章では, 実時間両方向探索 RTBS[Ishida 96] に多状態コミットメント [北村 99] の概念を導入した多状態実時間両方向探索 (MSRTBS : Multi-State Real-Time Bidirectional Search) を提案する . 前向き探索で生成された状態を保持するリストを $OPEN_1$, 計算用のリストを $TEMP_1$ とする . 同様に, 後向き探索で生成された状態を保持するリストを $OPEN_2$, 計算用のリストを $TEMP_2$ とする . 以下にアルゴリズムを示す .

< MSRTBS アルゴリズム >

Step1: 初期状態 s_0 を $OPEN_1$, 目標状態 G を $OPEN_2$ に加える . 保持する状態の数 n を設定する .

Step2: if ($OPEN_1 = \text{空}$ または $OPEN_2 = \text{空}$) then 探索失敗, 終了 .

Step3: $OPEN_1$ の先頭の状態を x , $OPEN_2$ の先頭の状態を y とする .

Step4: if ($x = y$) then 探索成功, 終了 .

Step5: if ($TEMP_1 = \text{空}$) then

- (1) $OPEN_1$ から先頭の状態 x を取り出す .
 (2) x を展開し生成される全ての子状態の集合と $OPEN_1$ に含まれる状態の集合を $TEMP_1$ に加える .
 (3) すべての $x' \in TEMP_1$ に関して $h(x', y)$ を計算する .

Step6: if ($TEMP_2 = \text{空}$) then

- (1) $OPEN_2$ から先頭の状態 y を取り出す .
 (2) y を展開し生成される全ての子状態の集合と $OPEN_2$ に含まれる状態の集合を $TEMP_2$ に加える .
 (3) すべての $y' \in TEMP_2$ に関して $h(x, y')$ を計算する .

Step7: 以下の式に従い, $h(x, y)$ を更新する .

$$h(x, y) = \text{secondmin}_{x', y'} \begin{cases} c(x, x') + h(x', y) \\ c(y, y') + h(x, y') \end{cases}$$

($h(x, y)$ の値を $c(x, x') + h(x', y)$ と $c(y, y') + h(x, y')$ のうち 2 番目に小さい値に更新する .)

Step8: 前向き探索と後向き探索の制御戦略

- (1) if ($\min\{h(x', y)\} < \min\{h(x, y')\}$) then
 a 全ての x' を $OPEN_1$ とし, 昇順にソートし, $TEMP_1$ を空にする (最小のものが複数個あれば最小のものをランダムに並び替える .)
 b if ($OPEN_1$ の個数 $> n$) then n 個になるまで, $OPEN_1$ の中で最大の評価値の状態を削除する .
 (2) if ($\min\{h(x', y)\} > \min\{h(x, y')\}$) then
 a 全ての y' を $OPEN_2$ とし, 昇順にソートし, $TEMP_2$ を空にする (最小のものが複数個あれば最小のものをランダムに並び替える .)
 b if ($OPEN_2$ の個数 $> n$) then n 個になるまで, $OPEN_2$ の中で最大の評価値の状態を削除する .
 (3) if ($\min\{h(x', y)\} = \min\{h(x, y')\}$) then ランダムで (1)(2) を実行する .

Step9: Step2 へ

4. 性能評価実験

提案した多状態実時間両方向探索 (MSRTBS) の性能を評価するために比較実験を行う . 比較対象は従来の多状態コミットメント探索 MSC-RTA* と両方向実時間探索 RTBS である . 評価対象としては, 探索の典型的な問題である N パズルと迷路を扱う . N パズル・迷路の両問題において, 各状態の評価値としては目標状態とのマンハッタン距離を用い, 状態 s から上下左右のそれぞれの状態 s' に移動させる 4 つのオペレータのコスト $c(s, s')$ を 1 とする .

なお, 本プログラムの実験環境は, オペレーティングシステム: Windows2000, 開発言語: Visual C++, CPU: Pentium4 2.4GHz, メモリ: 256MByte である .

表 1 N パズルにおける実験結果

サイズ N	手法	実行時間 (秒)	解の長さ	ステップ数	記憶節点数
24	MSC-RTA*	6.27	1326	9849	7451
	RTBS	5.49	1790	9403	5415
	MSRTBS	2.19	1475	6409	4030
35	MSC-RTA*	59.2	3834	23640	17911
	RTBS	53.4	4180	30021	15373
	MSRTBS	11.8	2875	14890	8044
48	MSC-RTA*	698	7600	82131	61827
	RTBS	331	8112	75712	36012
	MSRTBS	23	3549	20307	11040
63	MSC-RTA*	2422	9107	176071	78055
	RTBS	1713	10883	117278	88271
	MSRTBS	118	6564	43668	22162

表 2 迷路における実験結果

	実行時間 (秒)	解の長さ	実行ステップ数	記憶節点数
MSC-RTA*	150	503	8371	1089
RTBS	2248	527	27378	2788
MSRTBS	2606	496	33305	1811

4.1 実験 1: N パズル

N パズルは $\sqrt{N+1} \times \sqrt{N+1}$ の盤面に 1 から N までの番号のついたタイルと一つの空白があり、タイルをランダムな状態から、番号が正しく並び目標状態へとタイルをスライドさせてゆく問題である。

シミュレーション実験には、問題のサイズ N を 24, 35, 48, 63 として、ランダムに配置した問題において、アルゴリズム毎に各 80 回実行した。また、N パズルにおいて、MSC-RTA*の保持する状態数 n の数は 3 が最も性能が良いことが示されている [北村 99] ため、MSC-RTA*と MSRTBS の n の数は 3 とした。

それぞれのアルゴリズムにおける実行時間 (秒)、解の長さ、実行ステップ数、記憶節点数の平均値を表 1 に示す。なお、実行ステップ数とは問題解決器が状態を移動した回数であり、記憶節点数とは探索した問題空間の広さである。

実験結果より、本提案手法である MSRTBS は、従来の MSC-RTA*や RTBS と比べ、大幅に速度・解の精度が向上していることがわかる。これは、実行ステップ数と記憶節点数が少なくなったことから、探索空間が狭められたためであると考えられる。特に 63 パズルでは、MSC-RTA*の約 20 倍の速度向上、約 1.4 倍の解の精度向上、RTBS の約 15 倍の速度向上、約 1.7 倍の解の精度向上を実現している。

一方、MSC-RTA*は RTBS に比べて解の精度が良い。この理由は、多状態の MSC-RTA*では評価値が良い先祖の子状態を OPEN リストに保持しており、現在の状態の子状態より評価値が良い場合、先祖の子状態に移動し、解経路を修正することができるため、解経路を改善で

きたためである。同様に、MSRTBS においても、RTBS を多状態にすることで、解の精度は改善されていることがわかる。

4.2 実験 2: 迷路

迷路は入口 (0,0)、出口 (180,180) とする 180×180 の格子状グラフにおいて入口から出口への経路を求める問題である。グラフ内には通過することのできない障害物が 40% の割合でランダムに配置されている。なお、障害物の割合が多い迷路問題では、RTBS は有効に働かないことがわかっている [Ishida 96]。

MSC-RTA*、RTBS、MSRTBS のそれぞれのアルゴリズムにおいて 50 回実行した。実行時間 (秒)、解の長さ、実行ステップ数、記憶節点数の平均を表 2 に示す。

実験結果より、MSRTBS は MSC-RTA*と比べ、実行時間が約 15 倍低下されており、MSRTBS は RTBS と同様に迷路問題では有効に働かないことがわかる。この原因は、RTBS や MSRTBS は MSC-RTA*と比べ、実行ステップ数と記憶節点数が多いため、探索する問題の空間が広くなったためであると考えられる。

一方、MSRTBS が RTBS と比べて遅くなった原因として、記憶節点数が少なく、実行ステップ数が多いことから、探索する問題の空間が狭くなっているが、何度も同じ状態を探索しているためであると考えられる。しかしながら、同じ状態を何度も探索することで解を改善することができるため、解の質は MSRTBS が最も良いという結果が得られた。

5. ま と め

本研究では、実時間両方向探索に多状態コミットメントの概念を導入した MSRTBS (多状態実時間両方向探索) を提案した。そして、N パズルと迷路を用いて性能評価実験を行い、N パズルにおいて、従来の RTBS や MSC-RTA* と比べ、大幅に性能が改善されることを示した。今後の課題としては、理論的に提案手法の有効性を検証することである。

謝 辞

査読者の先生方には大変有益なコメントを頂きました。ここに記して深謝いたします。

◇ 参 考 文 献 ◇

- [Hart 68] Hart, P.E., Nilsson, N.J. and Raphael, B.: A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. and Cybern., Vol.4, No.2, pp.100-107 (1968).
- [Ishida 96] T.Ishida, "Real-time bidirectional search: coordinated problem solving in uncertain situations," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.18, No.6, pp.617-628 (1996).
- [北村 96] 北村泰彦, 寺西憲一, 辰巳昭治: マルチエージェント実時間探索における組織化とその評価, 人工知能誌, Vol.11, No.3, pp.470-477 (1996).
- [北村 99] 北村泰彦, 横尾 真, 宮地智久, 辰巳昭治: 多状態コミットメント探索とその評価, 人工知能誌, Vol.14, No.5, pp.860-869 (1999).
- [Korf 90] Korf, R.E.: Real-time heuristic search, Artificial Intelligence, Vol.42, pp.189-211 (1990).
- [Pohl 70] Pohl, I.: Heuristic search viewed as path finding in a graph, Artificial Intelligence, Vol.1, pp.193-204 (1970).
- [横尾 97] 横尾 真, 北村泰彦: 淘汰を用いたマルチエージェント実時間探索の高速化, コンピュータソフトウェア, Vol.14, No.4, pp.47-55 (1997).

〔担当委員: 北村泰彦〕

2003 年 8 月 25 日 受理

著 者 紹 介

越野 亮 (学生会員)

2000 金沢大学工学部電気情報工学科卒業。2002 同大学院博士前期課程修了。同年、富士通(株)入社。2003 金沢大学大学院博士後期課程数理情報科学専攻入学。石川工業高等専門学校電子情報工学科助手。現在に至る。この間、推論と探索の高速化の研究に従事。IEEE, AAAI, ACM/SIGART, 電子情報通信学会, 情報処理学会各会員

村田 裕章

2003 石川工業高等専門学校電子情報工学科卒業見込。人工知能の探索アルゴリズムに興味がある。電子情報通信学会会員

白山 政敏

1968 富山大学工学部生産機械卒業。1970 同大学院工学研究科修士課程生産機械工学専攻修了。工学博士。1983 富山工業高等専門学校機械工学科助教授。1989 石川工業高等専門学校電子情報工学科教授。現在に至る。その間、ニューラルネットワークや遺伝的プログラミングによる視覚移動ロボット制御に関する研究に従事。電子情報通信学会, 日本知識情報ファジィ学会, 精密工学会, 日本機械学会各会員

木村 春彦 (正会員)

1974 東京電機大学工学部応用理化学科卒業。1979 東北大学大学院工学研究科情報工学専攻博士課程修了。工学博士。同年、富士通(株)入社。1980 金沢女子短期大学講師。1984 金沢大学経済学部助教授。現在、同大学工学部情報システム工学科教授。この間、最適コード変換、プロダクションシステムの高速化の研究に従事。電子情報通信学会, 情報処理学会各会員