知能と複雑系 110－3
(1998. 1. 22)

ヒューリスティック探索への n-状態コミットメントの導入

宮地 智久†, 北村 泰彦†, 横尾 真††, 辰巳 昭治†
† 大阪市立大学工学部情報工学科
大阪市 住吉区 杉本 3-3-138
e-mail: {miya,kitamura,tatsumi}@kdel.info.eng.osaka-cu.ac.jp
†† NTT コミュニケーション科学研究所
京都府 相楽郡 精華町 光台 2-2
e-mail: yokoo@cslab.kecl.ntt.jp

あらまし

準最適解を求めるヒューリスティック探索の高速化のため，展開の候補を n-状態に限定する n-状態コミットメントの概念を導入する．対象とするヒューリスティック探索アルゴリズムとしては実時間 A*アルゴリズムと重み付き A*アルゴリズムであり，これらはそれぞれ 1-状態コミットメント，全状態コミットメント手法であると見なすことができる．2 分木を用いた理論的解析および迷路問題と n-パズル問題を用いた実験的解析でその効果を評価する．

キーワード　ヒューリスティック探索，コミットメント，準最適解

Introducing n-State Commitment Method Into Heuristic Searche Algorithms

Tomohisa MIYAJI†, Yasuhiko KITAMURA†, Makoto YOKOO††, Shoji TATSUMI†
† Dept. of Information and Communication Engineering,
Faculty of Engineering, Osaka City University
3-3-138 Sugimoto, Sumiyoshi-ku, Osaka 558 Japan
e-mail: {miya,kitamura,tatsumi}@kdel.info.eng.osaka-cu.ac.jp
†† NTT Communication Science Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan
e-mail: yokoo@cslab.kecl.ntt.jp

Abstract

To improve heuristic searche algorithms for semi-optimal solutions, we introduce n-state commitment method which limits the number of candidate states to be expanded to n. We apply the method to Weighted A* (WA*) algorithm and Real-Time A* (RTA*) algorithm which can be viewed as 1-state commitment and all-state commitment algrothms respectively, and show the effects through theoretical analysis using a binary tree and empirical analysis using maze and n-puzzle problems.

key words　heuristic search, commitment, semi-optimal solution

1 まえがき

人工知能の研究課題の一つに問題解決がある．ゲームやパズルは人間が行なう問題解決の典型的なものと考えられ，チェスや囲碁をするプログラムなどが数多く作られた．問題解決は，一般的に状態空間グラフにおける解経路を発見することに定式化することができ，このための手法として探索がある．しかしながら状態空間グラフのサイズが大きくなると単純な探索手法では現実的な時間内に解を求めることが困難になる．そこで問題に関するヒューリスティックな知識を状態評価関数として用いることによって性能を改善するヒューリスティック探索アルゴリズムが提案され，その代表的なものにA*アルゴリズムがある [Pearl84].

A*アルゴリズムは最適解が得られることを保証するが，実用的な探索アルゴリズムとしては，時間をかけて最適解を得ることよりも，許容可能な時間内に準最適解を得ることが望ましい場合も多い．そのようなヒューリスティック探索アルゴリズムとしては，重み付き A*(Weighted A*:WA*)[Korf93] アルゴリズムや実時間 A*(Real-Time A*:RTA*) アルゴリズム [Korf90] がある．

WA*と RTA*では展開する状態を決定する手法に特徴的な違いがある．WA*は探索木の最前線にある全ての状態を展開候補として記憶しているが，RTA*は先に展開した状態に隣接する状態のみに展開候補を絞り込んでいる．RTA*はヒューリスティック探索におけるコミットメントという視点からはその後の探索の可能性を展開する唯一の状態からの子孫に限定するという意味で 1-状態コミットメント探索アルゴリズムと見なすことができる．一方，WA*は探索の可能性を一切限定していないので，全状態コミットメント探索アルゴリズムと見なすことができる．[1]

さて，このようなコミットメント探索アルゴリズムは探索に関わるエージェントの数を増加させることにより，その性能が改善されることが報告されている．複数のエージェントが 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 A*アルゴリズム (Multiagent RTA*:MRTA*)Knight[Knight93] では n-パズルのような問題においてエージェント数の増加が性能向上につながることを示している．また横尾と北村は，MRTA*においてエージェントを再配置することで劇的な性能改善が可能であることを示した [横尾と北村 97].

コミットメントの視点からは MRTA*は n 台のエージェントのそれぞれが 1-状態コミットメント探索アルゴリズムを実行しているので，全体として n-状態コミットメント探索を行なっていると見なすことができる．一方で MRTA*では複数のエージェントが同一の状態を展開し，冗長な探索を行なう可能性が高くなるという問題点があった．そこで本研究ではマルチエージェントではなく，従来のコミットメント探索アルゴリズムを改良することによるシングルエージェント型の n-状態コミットメント探索アルゴリズムを提案する．すなわち，WA*, RTA*のそれぞれに最大長さ n のコミットメントリストを導入し，展開の候補はその中から選択するようにしている．

本稿では以下，ヒューリスティック探索の定式化を行なった後，WA*と RTA*アルゴリズムにおける n-状態コミットメント探索アルゴリズムへの修正法について述べる．また同時にこの修正を加えたとしてもアルゴリズムのもつ完全性の性質が変わらないことも示す．そして，n-状態コミットメント手法の効果を示すため，二分木を用いた理論的解析と，迷路問題と 15 パズルを用いた実験的解析を行ない，その性能を評価する．

2 ヒューリスティック探索の定式化

問題は四つ組 $\langle S, O, S_0, G\rangle$ で表される．ここで $S(\neq \phi)$ は状態の集合，Oは状態遷移を示すオペレータの集合，$s_0(\in S)$ は初期状態，$G(\subset S)$ は目標状態の集合である．問題は初期状態からオペレータを繰り返し適用して目標状態へ到達することによって解決される．

例として，図 1に迷路問題を示す．迷路問題はスタート (Start) からゴール (Goal) へ移動することによって解決される．したがって，迷路問題は障害物のないマスを状態，上下左右への移動をオペレータ，スタートを初期状態，ゴールを目標状態と見なすことで定式化できる．

もう一つの例として，図 2に 15 パズル問題を示す．15 パズル問題は n-パズル問題の一種であり，15 個の数字付きタイルとそれらを囲む枠からなる．問題は初期配置からタイルのスライドを繰り返し，目標とする配置に変更することによって解決される．n-パズル問題はタイルの配置が状態，タイルのスライドがオペレータ，タイルの初期配置が初期状態，目標とするタイルの配置が目標状態となる．

オペレータの適用にコスト (> 0) が定義される場合は，状態 s, $s'(\in S)$ 間の遷移のコストを $c(s, s')$ で示す．オペレータの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または解経路と呼ぶ．解経路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．ある解が存在したとき，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．

[1] 全ての状態にコミットすることは，どの状態にもコミットしていないこととも等価であるので，このアルゴリズムを無状態コミットメント探索アルゴリズムと見なすこともできる．

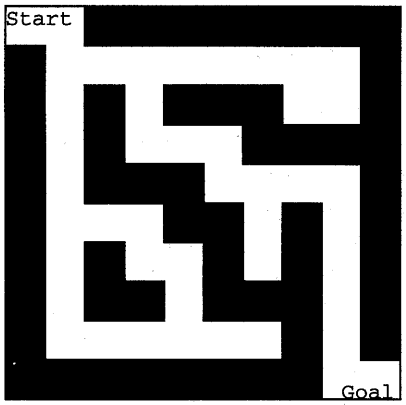

図 1: 迷路問題

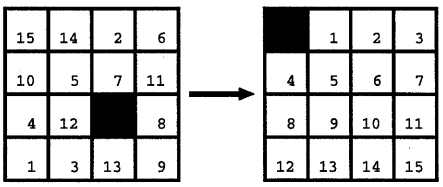

図 2: 15 パズル問題

ヒューリスティック探索では，各状態 $s(\in S)$ から目標状態までのコストの概算値を与えられており，ヒューリスティック関数 $h(s)$ で表される．

3 WA*アルゴリズムへの n−状態コミットメントの導入

3.1 WA*アルゴリズム

A*は状態 s の評価に初期状態から状態 s までのコストの和 $(g(s))$ と，状態 s から目標状態までのコストの概算値 $h(s)$ の和を用いるが，WA*は $g(s)$ と $h(s)$ の比重を変更できるアルゴリズムであり，状態 s の評価には $(1-W)g(s)+Wh(s)$ を用いる．このとき，Wが $h(s)$ に対する重みである．以下では，Wが極端に大きい場合を想定し，評価には $h(s)$ だけを用いる．

A*と同じくWA*もオープンリスト，クローズドリストと呼ばれる2つのリストを用いる．全ての生成された状態は一旦オープンリストに保存され，また一度展開された状態はクローズドリストに保存されるため，一度展開した状態は必ずいずれかのリストに存在する．従って，

$WA*(\langle S, O, s_0, G\rangle)$
1: $s \leftarrow s_0$;
2: generate $successors(s)$;
3: **if** $successors(s) \cap G \neq \phi$ **then return** $success$;
4: $add(open\_list, successors(s))$;
5: $add(closed\_list, s)$;
6: **if** $open\_list = \phi$ **then return** $failure$;
7: $s \leftarrow get\_min(open\_list)$;
8: **goto** 2;

図 3: WA*アルゴリズム

隣接する状態が生成済みか否かをいずれかのリストに存在するかどうかで判断できるため，WA*は状態の再生成を避けることができる．

WA*のアルゴリズムを図 3に示す．まず初期状態から探索を開始する (1 行目)．隣接する状態を全て生成する (2 行目)．ただし，過去に生成したことのある状態は生成しない．隣接する状態に目標状態が含まれていれば探索は成功したものとする (3 行目)．そうでなければ，隣接する状態をオープンリストに加え (4 行目)，展開した状態をクローズドリストに加える (5 行目)．もしオープンリストが空であれば探索は失敗したものとし (6 行目)，そうでなければ，オープンリストの中で最小の $h(s')(s' \in open\_list)$ を持つ状態を取り出し，探索を続行する (7,8 行目)．この一連の動作を 1 ステップと定義する．

[Korf90] で用いられている例 (図 4) で，WA*の動作を簡単に説明する．黒丸が状態，間をつなぐ枝がオペレータ，横の数値がもともとのヒューリスティック関数の値である．初期状態は a であり，オペレータのコストは全て 1 であるものとする．なお，この図では目標状態は示されていない．

まず，a に隣接する状態 b, c, d を生成する．いずれも目標状態ではないので，b, c, d をオープンリストに加え，a をクローズドリストに加える．オープンリストは空でないので，オープンリストから最小の評価値 $(h(b)=1)$ を持つ b を取り出す．ここまでが 1 ステップである．次に，b に隣接する状態 e, i を生成する (a は展開済みなので隣接する状態には含めない)．いずれも目標状態ではないので，e, i をオープンリストに加え，b をクローズドリストに加える．オープンリストは空でないので，オープンリストから最小の評価値 $(h(c)=2)$ を持つ c を取り出す．以下同様に c, d, e, i, ⋯と探索を行う．

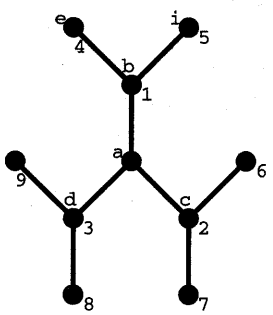

図 4: 例題

3.2 n-状態コミットメント WA*アルゴリズム

WA*はオープンリストに保存された状態から次に展開するものを選択する．n-状態コミットメントでは選択すべき候補をn状態に限定する．具体的には，コミットメントリストと呼ばれるリストを導入し，生成された状態はオープンリストではなくコミットメントリストに保存する．そして，コミットメントリストから次に探索を行う状態を選択する．コミットメントリストの長さはn以下に制限され，上限を越えた場合は評価の悪いものからオープンリストに移し変える．また，コミットメントリストが空になった場合は，オープンリストから評価の良いものを移し変える．なお，クローズドリストに関する扱いはWA*と同様である．従って，n–状態コミットメント WA*(n-state Commitment WA*:CWA*) でも一度生成した状態はコミットメントリスト，オープンリスト，クローズドリストのいずれかに必ず存在することが保証されている．CWA*のアルゴリズムを図5に示す．

WA*と比較すると，図3の4-7行目が図5の4-15行目に変更されている．

すなわち，生成した状態はオープンリストではなく，コミットメントリストに加える (4行目)．現在の状態はWA*と同様にクローズドリストに加える (5行目)．コミットメントリストが大きさの上限を越えていれば，評価値 $h(s')(s' \in commit\_list)$ が最大の状態からオープンリストに移し変える (6-8行目)．コミットメントリストに空きがあれば，コミットメントリストの上限までオープンリストから最小の $h(s')(s' \in open\_list)$ を持つ状態を移し変える (9-12行目)．コミットメントリスト，オープンリストともに空ならば探索に失敗に終る (13-14行目)．そして，オープンリストからではなく，コミットメントリストの中から次に探索を行う状態を選択する (15行目)．

$CWA*(\langle S, O, s_0, G\rangle)$
1: $s \leftarrow s_0$;
2: generate $successors(s)$;
3: **if** $successors(s) \cap G \neq \phi$ **then return** $success$;
4: $add(commit\_list, successors(s))$;
5: $add(closed\_list, s)$;
6: **while** $length(commit\_list) > n$ **do**
7: $s \leftarrow get\_max(commit\_list)$;
8: $add(open\_list, s)$;
9: **while** $(length(commit\_list) < n)$ **and**
10: $(open\_list \neq \phi)$ **do**
11: $s \leftarrow get\_min(open\_list)$;
12: $add(commit\_list, s)$;
13: **if** $(commit\_list = \phi)$ **and** $(open\_list = \phi)$ **then**
14: **return** $failure$;
15: $s \leftarrow get\_min(commit\_list)$;
16: **goto** 2;

図 5: CWA*アルゴリズム

コミットメントリストの大きさの上限 (n) を無限大にすると，もともとのWA*と等しくなる．

3.3 アルゴリズムの完全性

WA*に n–状態コミットメントを導入しても完全性が保たれていることを示す．なお，この場合の完全性とは，初期状態から目標状態に至る経路が存在するとき，解経路が得て，停止することを保証するものである．

WA*の完全性は，初期状態から目標状態への経路が存在し，状態空間が有限であるとき保証されている．これはWA*は一度展開した状態を再展開しないため無限ループに陥ることがなく，また，アルゴリズムが停止する以前では解経路上の状態が少なくとも一つはオープンリストに保存されているからである．

CWA*も同様に，状態を再展開することがないので無限ループには陥らず，解経路上の状態は少なくとも一つコミットメントリストあるいはオープンリストに存在する．従って，初期状態から目標状態への経路が存在し，状態空間が有限ならば，CWA*の完全性は保証される．

$RTA*(\langle S, O, s_0, G\rangle)$
1: $s \leftarrow s_0$;
2: generate $successors(s)$;
3: **if** $successors(s) \cap G \neq \phi$ **then return** $success$;
4: $update(s)$;
5: **if** $successors(s) = \phi$ **then return** $failure$;
6: $s \leftarrow get\_min(successors(s))$;
7: **goto** 2;

図 6: RTA*アルゴリズム

4 RTA*アルゴリズムへのn−状態コミットメントの導入

4.1 RTA*アルゴリズム

Korf の提案した RTA*は，探索しながらヒューリスティック関数の値を更新することが特徴的である．RTA*のアルゴリズムを図 6に示す．

以下にアルゴリズムの流れを説明する．まず初期状態から探索を開始する (1 行目)．そして現在の状態に隣接する状態を全て生成する (2 行目)．ただし，状態の評価値が無限大 ($h(s') = \infty (s' \in successors(s))$) の場合，その状態より先に探索を進めても目標状態へ到達することはないので，隣接する状態に含めない．隣接する状態に目標状態が含まれていれば探索は成功である (3 行目)．目標状態が含まれていなければ，$h(s)$ を隣接する状態の中で 2 番目に小さい $c(s, s') + h(s')(s' \in successors(s))$ に更新する (4 行目)．隣接する状態の数が 1 個以下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．もし評価値が ∞ でない隣接する状態が存在しなければ探索は失敗である (5 行目)．そうでなければ，隣接する状態の中で最小の評価値 $h(s')(s' \in successors(s))$ を持つ状態を次に展開する (6,7 行目)．なお，複数の状態の評価値が同じときはその中からランダムに選択する．

図 4で RTA*の動作を簡単に説明する．まず，a に隣接する状態 b, c, d を生成する．いずれも目標状態ではないため，$h(a)$ を更新する．この場合 $c(a,b)+h(b) = 1+1 = 2$, $c(a,c)+h(c) = 1+2 = 3$, $c(a,d)+h(d) = 1+3 = 4$ となるので，$h(a)$ は 2 番目に小さい 3 に更新される．評価値が ∞ でない隣接する状態は存在するので，最小の評価値 ($h(b) = 1$) を持つ b を次に展開する．次に，b に隣接する状態 a, e, i を生成する．いずれも目標状態ではないため，$h(b)$ を更新する．この場合 $c(b,a)+h(a) = 1+3 = 4$, $c(b,e)+h(e) = 1+4 = 5$, $c(b,i)+h(i) = 1+5 = 6$ となるので，$h(b)$ は 2 番目に小さい 5 に更新される．そして，最小の評価値 $h(a) = 3$ を持つ a を次に展開する．以下同様に評価値を更新しつつ，a から c, c から a, a から d, ···と探索が行われる．

$CRTA*(\langle S, O, s_0, G\rangle)$
1: $s \leftarrow s_0$;
2: generate $successors(s)$;
3: **if** $successors(s) \cap G \neq \phi$ **then return** $success$;
4: $update(s)$;
5: $add(commit\_list, successors(s))$;
6: **while** $length(commit\_list) > n$ **do**
7: $s \leftarrow get\_max(commit\_list)$;
8: **if** $commit\_list = \phi$ **then return** $failure$;
9: $s \leftarrow get\_min(commit\_list)$;
10: **goto** 2;

図 7: CRTA*アルゴリズム

4.2 n-状態コミットメント RTA*アルゴリズム

RTA*は，展開された状態の子孫のみが以後の探索に影響を与える 1-状態コミットメント探索アルゴリズムである．ここでは，RTA*に n−状態コミットメントの概念を導入することにより，コミットメントの範囲を広げる．例えば，先程の例で b から次に探索を行う状態を選択する際，隣接する a, e, i だけではなく c も候補に入れられ，a を展開ではなく c が展開される．

具体的には，CWA*と同様にコミットメントリストを導入し，生成した状態をコミットメントリストに保存する．そして，コミットメントリストから次に展開を行う状態を選択する．コミットメントリストの長さは n に制限され，それを越えた場合は評価値の悪いものから取り除く．

n−状態コミットメント RTA*(n-state Commitment RTA*:CRTA*) アルゴリズムを図 7に示す．

RTA*のアルゴリズムと比較すると，図 6の 5,6 行目が図 7の 5-8 行目に変更されている．すなわち，隣接する状態は一旦コミットメントリストに加えられる (5 行目)．ただし，コミットメントリスト内で同じ状態の重複は許さない．コミットメントリストの大きさが上限を越えていれば，評価値 $h(s')(s' \in commit\_list)$ が最大のものから削除する (6,7 行目)．もしコミットメントリストが空ならば，探索は失敗に終る (8 行目)．展開はコミットメントリストに保存されている状態から選択する (9 行目)．なお，コミットメントリストから状態を取り出す際，同じ評価値のものがあればランダムに選択する．

コミットメントリストの大きさの上限(n)を1とすると，もともとのRTA*と等しくなる．

4.3 アルゴリズムの完全性

RTA*の完全性は全ての状態から目標状態に到達可能，ヒューリスティック関数の初期値は有限，状態空間が有限の場合に保証されている [Korf90]．

以下では，RTA*にn−状態コミットメントを導入しても，完全性が保たれていることを示す．まず，RTA*と同じく，各状態に対するヒューリスティック関数の初期値が有限であるので，無限なヒューリスティック値の障害が目標を取り囲んでアルゴリズムが失敗で終了することはない．

また，全ての状態から目標状態への経路が存在し，状態空間が有限であるので，無限のループに陥ることはない．その理由は以下からなる．状態空間が有限で目標状態への経路が存在するにもかかわらず，目標状態へ到達できないならば，目標状態を含まない無限ループに陥っていると考えられる．状態の評価値は隣接する状態の評価値とオペレータのコストの和に更新されるため，更新された評価値は隣接する状態の評価値よりも必ず大きくなる．よって，ループ内における最小の評価値は一巡する毎に確実に増加する．したがってループに含まれる状態の最小の評価値は，いずれそれに含まれない状態の評価値よりも大きくなり，ループから抜け出すことができる．状態の数は有限であるので，いずれ目標状態にも到達する．この性質は，コミットメントリストの長さに関わらず保たれ，CRTA* の完全性は保証される．

5 理論的解析

5.1 CWA*

各状態において左の子供の方が右の子供よりも評価値が良いように作られた二分木を考える．そのような二分木の探索において，左右いずれか一方の部分木にしか目標状態が存在しないような問題を仮定する．この一例として図8に示すようなs_0を根とする木を考える．s_0は左の子供s_1と右の子供s_2を持ち，また，それぞれの子供もs_1を根とする部分木$T(s_1)$，s_2を根とする部分木$T(s_2)$を持つ．

s_0からs_1とs_2が生成された後，s_1の探索が行われる．以下では，このような問題において，n−状態コミットメントの効果が次の2点に大きく依存していることを示す．

- $T(s_1)$ 中で最大の $h(s)(s \in T(s_1))$ が $h(s_2)$ より小さいか．すなわち $\max_{s \in T(s_1)} h(s) < h(s_2)$.

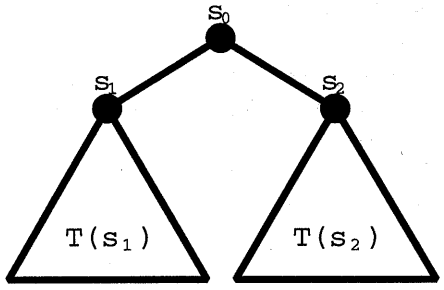

図 8: 二分木

- $T(s_1)$ に目標状態が含まれるか否か．

まず，$\max_{s \in T(s_1)} h(s) < h(s_2)$ となる場合を考える．CWA*はコミットメントリストの中で最小の $h(s)(s \in commit\_list)$ を持つ状態から探索を行うので，n がいくらであっても，$T(s_1)$ の探索が終了するまで s_2が展開されることはない．従って，$T(s_1)$ の探索が終了するまで $T(s_2)$ が探索されることはなく，$T(s_1)$ と $T(s_2)$ のいずれに解が含まれていようとも n−状態コミットメントの効果は期待できない．

次に，$\max_{s \in T(s_1)} h(s) \geq h(s_2)$ かつ，$G \subset T(s_1)$ なる場合を考える．この場合，n が大きいと $T(s_1)$ の探索によって目標状態に到達する前に s_2がコミットメントリストから選択されて余分な探索を行う可能性がある．WA*はCWA*における $n = \infty$ の場合と等しいので，n を小さくすれば，CWA*がWA*よりも少ないステップ数で解を得ることを期待できる．

最後に，$\max_{s \in T(s_1)} h(s) \geq h(s_2)$ かつ，$G \subset T(s_2)$ なる場合を考える．$T(s_1)$ の探索を極力減らし，できるだけ早く $T(s_2)$ で解を見つけたい状況である．n が大きければ，$T(s_1)$ の探索が終了する前に s_2がコミットメントリストから選択されて $T(s_2)$ の探索が開始される可能性がある．これはWA*の方がCWA*よりも適している場合である．

5.2 CRTA*

まず $\max_{s \in T(s_1)} h(s) < h(s_2)$ なる場合を考える．CWA*と同様に，CRTA*もコミットメントリストの中で最小の $h(s)(s \in commit\_list)$ を持つ状態から探索を行うので，n がいくらであっても，$T(s_1)$ の探索が終了するまで s_2の探索が行われることはない．従って，$T(s_1)$ の探索が終了するまで $T(s_2)$ が探索されることはなく，$T(s_1)$ と $T(s_2)$ のいずれに解が含まれていようとも n−状態コミットメントの効果は期待できない．

表 1: CWA*の迷路問題における実験結果

アルゴリズム	成功率	ステップ数	解の長さ
CWA* (1)	100	391.7	170.8
CWA* (2)	100	383.0	167.7
CWA* (4)	100	383.0	167.3
CWA* (8)	100	383.9	167.5
CWA*(12)	100	384.2	167.6
WA*	100	383.9	167.4

表 2: CWA*の 15 パズル問題における実験結果

アルゴリズム	成功率	ステップ数	解の長さ
CWA* (1)	100	3940.7	1188.8
CWA* (2)	100	2274.0	1568.7
CWA* (4)	100	1578.2	760.9
CWA* (8)	100	1576.2	529.5
CWA*(12)	100	1694.4	463.6
WA*	100	1889.0	203.6

次に，$\max_{s \in T(s_1)} h(s) \geq h(s_2)$ かつ，$G \subset T(s_1)$ なる場合を考える．このような状況では n はあまり大きくない方が適している．そして，RTA*はCRTA*において $n = 1$ の場合と等しい．従って，このような状況では，RTA*の方がCRTA*よりも適している．

最後に，$\max_{s \in T(s_1)} h(s) \geq h(s_2)$ かつ，$G \subset T(s_2)$ なる場合を考える．この場合，コミットメントリストの上限が大きい方が適しているため，n–状態コミットメントの効果が期待できる．

表 3: CWA*の 48 パズル問題における実験結果

アルゴリズム	成功率	ステップ数	解の長さ
CWA* (1)	38	370374.0	9495.0
CWA* (2)	97	190507.0	9470.7
CWA* (3)	98	134848.0	10180.7
CWA* (4)	100	120051.0	11138.1
CWA* (5)	96	133266.0	11584.8
CWA* (6)	99	157416.0	11622.1
WA*	10	189241.9	3151.0

6 実験的解析

6.1 CWA*

まず，CWA*を用いて迷路問題上で実験を行った．障害物の比率を 40%とし，入口 (0,0)，出口 (59,59) とする 60x60 の格子状グラフを用いた．各状態のヒューリスティック関数には目標状態とのマンハッタン距離を用い，各オペレータのコストは 1 とした．ランダムに生成した 100 個の迷路問題に対し各 100 回試行した結果を表 1に示す．括弧内の数字は n である．成功率は実験を行った問題のうち，何パーセントの問題で解を得ることができたかを示す．

迷路問題では n を増やしても性能にほとんど変化がなく，WA*と比較しても性能の改善は得られなかった．

次に，CWA*を用いて 15 パズル問題上で実験を行った．各状態のヒューリスティック関数には各タイルの正しい位置とのマンハッタン距離の和を用い，各オペレータのコストは 1 とした．ランダムに生成した 100 個の 15 パズルに対し各 100 回試行した結果を表 2に示す．

15 パズル問題上で CWA*を実行すると，$n = 8$ としたとき最も少ないステップ数で解を得ることができた．n を増やすと最終的には WA*の結果に収束するものと思われる．解の長さに関しては，n の増加に対して単調減少しており，結果として WA*より良い解を得ることはできなかった．

また，48 パズル問題でも実験を行った．ヒューリスティック関数などは 15 パズル問題と同様である．ランダムに生成した 100 個の 48 パズルに対し各 1 回試行した結果を表 3に示す．なお，実験に使用した計算機には利用可能なメモリ量に限界があるため，各リストに含まれる状態数の和が 150 万を越えた場合，探索が失敗したものとみなしている．

48 パズル問題では，WA*ではわずか 1 割の問題でしか解を得ることができなかったのに対し，$n = 4$ としたときのCWA*は全ての問題で解を得ることができた．ステップ数で比較すると，$n = 4$ の場合，WA*の 2/3 のステップ数で解を得ることができた．解の長さに関しては WA*が極端に良いように見えるが，これは解の短いものしか発見できないことを表していると考えることもできる．

6.2 CRTA*

CWA*と同様の条件で迷路問題，15 パズル問題上で CRTA*を実行した．迷路問題の結果を表 4に，15 パズル問題の結果を表 5に示す．

迷路問題では，CRTA*と同様に n を変化させても結果はほとんど変わらず，RTA*と比較しても性能の改善は得られなかった．

15 パズルでは，$n = 2$ としたときステップ数が最小と

表 4: CRTA*の迷路問題における実験結果

アルゴリズム	成功率	ステップ数	解の長さ
RTA*	100	1340.2	165.1
CRTA* (2)	100	1352.4	163.3
CRTA* (4)	100	1352.7	163.2
CRTA* (8)	100	1350.3	163.4
CRTA*(12)	100	1318.9	163.4

表 5: CRTA*の 15 パズル問題における実験結果

アルゴリズム	成功率	ステップ数	解の長さ
RTA*	100	2647.1	1049.3
CRTA* (2)	100	1380.1	447.1
CRTA* (4)	100	1442.4	358.4
CRTA* (8)	100	1712.5	334.9
CRTA*(12)	100	1812.7	326.1

なり，それ以上に n を増やすとステップ数は単調に増加した．また，解の長さは n を増やす程短くなった．RTA*と比較すると，$n=2$ としたとき，CRTA*はRTA*の約半分のステップ数で解を見つけることができており，解の長さも半分以下になっている．よって，15 パズル問題において，RTA*への n–状態コミットメントの導入は効果的であることが示された．

48 パズル問題の結果を図 6に示す．$n=3$ としたときステップ数が最小となり，それ以上に n を増やすとステップ数は単調に増加した．RTA*と比較すると，$n=3$ としたとき，CRTA*はRTA*の 1/4 以下のステップ数で解を見つけることができており，解の長さは約 1/10 になっている．従って，48 パズル問題においても，RTA*への n–状態コミットメントの導入は効果的であることが示された．

以上の実験結果から，迷路問題では理論的解析において n–状態コミットメントの効果が期待できないと考えられる $\max_{s\in T(s_1)} < h(s_2)$ を満たす状況がほとんどだと考えられる．

同様に，15 パズル問題および 48 パズル問題には，$\max_{s\in T(s_1)} \geq h(s_2)$ かつ $G \subset T(s_2)$ を満たす状況と $\max_{s\in T(s_1)} \geq h(s_2)$ かつ $G \subset T(s_1)$ を満たす状況が多く含まれており，前者の状況の方がより多いため，CWA*よりも CRTA*において n–状態コミットメントの効果がはっきり現れたと考えられる．

表 6: CRTA*の 48 パズル問題における実験結果

アルゴリズム	成功率	ステップ数	解の長さ
RTA*	2	393439.0	228849.0
CRTA* (2)	100	155770.0	44569.0
CRTA* (3)	100	93822.0	23426.0
CRTA* (4)	100	120640.0	21231.0
CRTA* (5)	100	181778.0	24243.0
CRTA* (6)	100	210497.0	24512.0

7 むすび

本研究では，探索の範囲を制限する n–状態コミットメントの概念を WA*と RTA*の二つのヒューリティック探索アルゴリズムに導入した．そして，二分木を用いた理論的解析により，n–状態コミットメントを導入したアルゴリズムがどのような状況で効果的かを示した．

また，迷路問題，15 パズル問題および 48 パズル問題上で実験的解析を行った．いずれのアルゴリズムでも迷路問題では性能を改善できなかったが，n-パズル問題では性能を改善できた．特に 48 パズル問題において CRTA*は RTA*の 1/4 以下のステップ数で，CWA*は WA*の 2/3 のステップ数で解を得ることができた．

今後の課題としては，理論的解析をより厳密に行い，n–状態コミットメントの効果を明確にすることが考えられる．

参考文献

[Knight93] K.Knight: "Are Many Reactive Agents Better Than a Few Deliberative Ones?," In *Proc. of IJCAI-93*, pp.432-437(1993).

[Korf90] R.E.Korf: "Real-Time Heuristic Search," *Artificial Intelligence*, Vol.42,No.2-3(1990),pp.189-211.

[Korf93] R.E.Korf: "Liner-space best-first search," *Artificial Intelligence*, Vol.62,No.1(1993),pp.41-78.

[Pearl84] J. Pearl: "Heuristics," Addison Wesley (1984).

[横尾と北村 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:協調探索への競争の導入," コンピュータソフトウェア, Vol.14, No.4(1997),pp.47-55.