論 文 Technical Papers

多状態コミットメント探索とその評価

Multi-State Commitment Search and Its Evaluation

北村 泰彦 *1 Yasuhiko Kitamura　横尾 真 *2 Makoto Yokoo　宮地 智久 *1,† Tomohisa Miyaji　辰巳 昭治 *1 Shoji Tatsumi

*1 大阪市立大学工学部情報工学科
Dept. of Information and Communication Engineering, Faculty of Engineering, Osaka City University, Osaka 558-8585, Japan.
*2 NTTコミュニケーション科学研究所
NTT Communication Science Laboratories, Kyoto 619-0237, Japan.

19YY年MM月DD日 受理

Keywords: heuristic search, RTA*, weighted A*, commitment

Summary

We propose the Multi-State Commitment (MSC) method to speed up heuristic search algorithms for semi-optimal solutions. The Real-Time A* (RTA*) and the Weighted A* (WA*) are representative heuristic search algorithms for semi-optimal solutions and can be viewed as a single-state and an all-state commitment search algorithms respectively. In these algorithms, there is a tradeoff between the risk of making wrong choices in search process and the amount of memory for the recovery, with RTA* and WA* being the extremes. The MSC method introduces a moderate and flexible characteristic into these algorithms and can increase the performance dramatically in problems such as the N-puzzle. In this paper, by introducing a commitment list, we show a modification of RTA* and WA* to their MSC versions without violating their completeness. Then, we experiment with their performance in maze and N-puzzle problems, and discuss conditions that the MSC method is effective.

1. まえがき

人工知能における問題解決は，一般に状態空間グラフにおける初期状態から目標状態までの解経路を発見することとして定式化することができ，その問題解決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[Pearl 84]．

A*アルゴリズムは最適解を発見す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*)[Korf 93]アルゴリズムや実時間A*(Real-Time A*:RTA*)アルゴリズム[Korf 90]がある．

WA*とRTA*では展開する状態を選択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の子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次に展開する単一の状態からの子孫に限定するという意味で単一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

† 現在，住友電気工業（株）

*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

準最適解探索は一般に，展開する状態を一つ以上の候補の中から選択する判断を繰返し行なうが，選択の誤りは目標状態までの探索ステップを長くしてしまうという悪影響を及ぼす．したがってRTA*のように以後の探索の可能性を単一の状態の子孫に絞り込んでしまうアルゴリズムでは，その判断を誤った場合における悪影響は大きい．一方，選択を誤った場合における回復を容易にするためにはWA* のように可能な候補を全てオープンリストとして保存しておくという方法もある．しかしこの方法では探索の範囲が広くなりすぎて計算機の記憶領域をすぐに消費してしまい，実質的に解が得られないという問題がある．このように探索における選択誤りのリスクと回復のための記憶量の間にはトレードオフの関係があり，そのそれぞれの両極端が単一状態コミットメント探索アルゴリズムのRTA*と，全状態コミットメント探索アルゴリズムのWA*であるといえる．

本論文では単一状態コミットメントと全状態コミットメントの中間的なものとしてパラメータnをもつ多状態コミットメント探索アルゴリズムを提案する．これは選択誤りリスクと回復に必要な記憶量のバランスを柔軟にとることができ，うまくnの値を調整すれば，N-パズルのような問題に対しては極めて効果的であることを示す．本論文では以下，ヒューリスティック探索の定式化を行なった後，WA*とRTA*アルゴリズムの多状態コミットメント探索アルゴリズムへの拡張法について述べる．また同時にこの拡張を加えたとしても，もとのアルゴリズムのもつ，解を発見し停止するという完全性が損なわれないことも示す．そして，多状態コミットメント手法の効果を示すため，迷路とN-パズルを用いたシミュレーション実験を行ない，N-パズルに関しては劇的な性能向上が見られることを示す．最後に，多状態コミットメント探索が有効な問題に関する考察と関連研究を示す．

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．ここで組$\langle S, O\rangle$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(> 0)が定義される場合は，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$s(\in S)$から目標状態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．とくにこの推測値が実際の値を上回らない場合，そのヒューリスティック関数は適格(admissible)であるといわれる．

3. WA*アルゴリズムへの多状態コミットメントの導入

3・1 WA*アルゴリズム

A*アルゴリズムは状態sの評価に初期状態から状態sまでのコストの推測値$g(s)$と，状態sから目標状態までのコストの推測値$h(s)$の和を用いるが，WA*は$g(s)$と$h(s)$の比重を変化させたもので，状態sの評価には$(1-W)g(s)+Wh(s)$を用いる．ただし$W(\in \mathbf{R})$は$0 \leq W \leq 1$である．一般にWの値が大きいほど解を早く発見することができるが，逆にその質は悪くなる．本稿では準最適解発見の高速化に興味があるので，以下では$W = 1$の場合を議論する．

WA*はA*と同じくOPEN，CLOSEDと呼ばれる2つのリストを用いる．生成された状態は一旦OPENに保存され，また展開された状態はCLOSEDに移されるため，一度生成された状態は必ずいずれかのリストに存在する．従って，それらのリストをチェックすることでWA*は状態の再生成を避けている．*2

WA*アルゴリズムを図1に示す．まず初期状態から探索を開始する(1行目)．状態を展開し，子状態を生成する(2行目)．ただし，OPENあるいはCLOSEDに入っている状態は生成しない．生成した状態に目標状態が含まれていれば探索は成功で終了である(3行目)．そうでなければ，生成した状態をOPENに加え(4行

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．

*2 解の質を考慮する場合は状態の評価値が前のものよりもよければ状態を再生成する（CLOSEDからOPENに移動する）ことがあるが，ここでは$W = 1$の場合を扱い，$f(s)(= h(s))$の値は固定されているので，そのような操作は行なわれない．

```
WA*
1:  s ← s_0;
2:  generate successors(s);
3:  if successors(s) ∩ G ≠ φ then return success;
4:  add(OPEN, successors(s));
5:  add(CLOSED, s);
6:  if OPEN = φ then return failure;
7:  s ← get_min(OPEN);
8:  goto 2;
```

図1 WA*アルゴリズム

```
MSC-WA*
1:  s ← s_0;
2:  generate successors(s);
3:  if successors(s) ∩ G ≠ φ then return success;
4:  add(COMMITMENT, successors(s));
5:  add(CLOSED, s);
6:  while length(COMMITMENT) > n do
7:      s ← get_max(COMMITMENT);
8:      add(OPEN, s);
9:  while (length(COMMITMENT) < n) and
10:         (OPEN ≠ φ) do
11:     s ← get_min(OPEN);
12:     add(COMMITMENT, s);
13: if (COMMITMENT = φ) then return failure;
14: s ← get_min(COMMITMENT);
15: goto 2;
```

図2 MSC-WA*アルゴリズム

目)，展開した状態をCLOSEDに加える(5行目)．もしOPENが空であれば探索は失敗であり(6行目)，そうでなければ，OPENの中で最小の評価値$h(s)$を持つ状態sを取り出し(7行目)，探索を続ける(8行目)．この一連の動作を1ステップと定義する．

3・2 MSC-WA*アルゴリズム

WA*はOPENに保存された状態から次に展開するものを選択する全状態コミットメント探索アルゴリズムである．多状態コミットメントWA*(MSC-WA*)では選択すべき候補をn状態に限定する．そのために，COMMITMENTと呼ばれる新たなリストを導入し，生成された状態はOPENではなくCOMMITMENTに追加する．そして，COMMITMENTから次に展開を行う状態を選択する．COMMITMENTの長さはn以下に制限され，上限を越えた場合は評価値の悪いものからOPENに移し変える．また，COMMITMENTの長さがn未満になった場合は，OPENから評価値の良いものを移し変える．なお，CLOSEDに関する扱いはWA*と同様である．従って，MSC-WA*においても一度生成した状態はCOMMITMENT, OPEN, CLOSEDのいずれかに必ず存在する．MSC-WA*のアルゴリズムを図2に示す．

MSC-WA*において子状態はCOMMITMENT, OPEN, CLOSEDに含まれないもののみを生成する(2行目)．また，生成した状態はOPENではなく，COMMITMENTに加える(4行目)．展開した状態はWA*と同様にCLOSEDに加える(5行目)．COMMITMENTが長さの上限を越えていれば，評価値が最大の状態からOPENに移し変える(6-8行目)．COMMITMENTに空きがあれば，COMMITMENTの上限までOPENから最小の評価値を持つ状態を移し変える(9-12行目)．COMMITMENTが空ならば探索は失敗に終る(13行目)．そして，次に展開を行う状態としてCOMMITMENTの中から評価値最小の状態を選択する(14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．

MSC-WA*は$n = \infty$のときに，もとのWA*と等しくなる．

3・3 アルゴリズムの性質

まず，WA*に多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，アルゴリズムが必ず解を得て停止することである．

WA*の完全性は，初期状態から目標状態への経路が存在し，状態空間グラフが有限であるときに保証されている．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OPENに保存されているからである[Shirai 82]．

MSC-WA*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必ず停止する．失敗で終了するのはCOMMITMENTが空の場合(13行目)であるが，この場合には9-12行目の操作によりOPENもともに空になっているはずである．しかし解経路上の状態は少なくとも一つはCOMMITMENTあるいはOPENに存在するので，失敗で終了することはない．以上より，初期状態から目標状態への経路が存在し，状態空間グラフが有限ならば，MSC-WA*の完全性は保証される．

次にMSC-WA*で得られる解の最適性について説明を加えよう．先述したようにWA* は状態sの評価値として$(1-W)g(s)+Wh(s)$を用いており，$h(s)$が適格で，$0 \leq W \leq 1/2$の場合には最適解を得ることが保証される．また$1/2 < W \leq 1$の場合には最適解のコストをC^*とした時，得られる解のコストCは$C^* \leq C \leq C^*(1+\varepsilon)$となることが保証される．ここで

```
RTA*
1:  s ← s_0;
2:  generate successors(s);
3:  if successors(s) ∩ G ≠ φ then return success;
4:  update_h(s);
5:  if successors(s) = φ then return failure;
6:  s ← get_min(successors(s));
7:  goto 2;
```

図3 RTA*アルゴリズム

$\varepsilon = W/(1-W)-1$である．*3しかしながらMSC-WA*では展開候補はOPENリストとCOMMITMENTリストに分断されてしまい，$n=\infty$の場合を除いて，評価値最小の状態が展開の対象となるCOMMITMENTリストに含まれるという保証がないので，解のコストの保証もない．

4. RTA*アルゴリズムへの多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*[Korf 90]は先読み探索と移動を交互に行なう準最適解探索アルゴリズムである．*4先読み深さを大きくすれば解の質を改善することができるが，ここでは解発見の高速化に焦点をあて，その深さは1の場合のみを扱う．またアルゴリズムの完全性を保証するために，RTA*ではCLOSEDリストを用いず，探索しながら評価値を更新してゆく．RTA*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)．そして子状態の全てを生成する(2行目)．ただし，子状態$s'(\in successors(s))$の評価値が無限大$(h(s')=\infty)$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，生成状態に含めない．生成した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，$h(s)$を子状態の中で2番目に小さい$c(s,s')+h(s')(s' \in successors(s))$に更新する(4行目)．子状態の数が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∞でない子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，子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．

*3 $W=1$の場合は$\varepsilon=\infty$となり得られる解のコストの保証はない．

*4 RTA*アルゴリズムには二つの解釈がある．一つの解釈はRTA*が準最適解を求めるためのオフラインアルゴリズムで，もう一つの解釈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オンラインアルゴリズムであるというものである．本論文では前者の解釈をとっており，多状態コミットメントの導入によりその性能を改善しようとしている．

```
MSC-RTA*
1:  s ← s_0;
2:  generate successors(s);
3:  if successors(s) ∩ G ≠ φ then return success;
4:  update_h(s);
5:  add(COMMITMENT, successors(s));
6:  while length(COMMITMENT) > n do
7:      remove_max(COMMITMENT);
8:  if COMMITMENT = φ then return failure;
9:  s ← get_min(COMMITMENT);
10: goto 2;
```

図4 MSC-RTA*アルゴリズム

4・2 MSC-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単一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*に多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．すなわち，MSC-WA*と同様にCOMMITMENTリストを導入し，生成した状態をそのリストに保存する．そして，そこから次に展開を行う状態を選択する．COMMITMENTの長さはn以下に制限され，それを越えた場合は評価値の悪いものから取り除く．多状態コミットメントRTA*(Multi-State Commitment RTA*: MSC-RTA*)アルゴリズムを図4に示す．

RTA*アルゴリズムと比較すると，図3の5,6行目が図4の5-8行目に変更されている．すなわち，生成した状態は一旦COMMITMENTに加えられる(5行目)．ただし，COMMITMENT内で同じ状態の重複は許さない．COMMITMENTの長さが上限を越えていれば，評価値$h(s)$が大きいものから削除する(6,7行目)．もしCOMMITMENTが空ならば，探索は失敗に終る(8行目)．次に展開する状態はCOMMITMENTに保存されているものから評価値が最小のものを選択する(9行目)．なお，COMMITMENT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．

MSC-RTA*は$n=1$のとき，もとのRTA*と等しくなる．

4・3 アルゴリズムの性質

RTA*の完全性は初期状態から到達可能な全ての状態から目標状態に到達可能，評価値の初期値が有限，状態空間が有限の場合に保証されている[Korf 90]．

以下では，RTA*を多状態コミットメントRTA*に

変更しても，完全性が保たれていることを示す．まず，RTA*と同じく，各状態に対する評価値の初期値が有限であり，初期状態から目標状態に至る経路は存在するので，評価値が無限大となる状態が目標状態を取り囲んでアルゴリズムが目標状態まで到達できずに失敗で終了することはない．

また，以下の理由によりアルゴリズムは無限ループに陥ることもない．もしアルゴリズムが無限ループに陥っているとすれば，そこには当然のことながら目標状態は含まれない．展開された状態の評価値は子状態の評価値と非負のオペレータのコストの和の2番目に良いものに更新されるため，更新された評価値は子状態の評価値の最小のものよりも必ず大きい．よって，ループ内の状態の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になる．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nに依存しない．ゆえに，MSC-RTA*の完全性は保証される．

次に，MSC-RTA*で得られる解の最適性に関して述べる．RTA*アルゴリズムは初期状態からの状態sへのコストを示す$g(s)$を考慮しておらず，解のコストの保証はない．したがって，MSC-RTA*においても解の最適性は保証されない．

5. 性能評価実験

多状態コミットメント手法の性能を評価するための実験をおこなった．用いた問題は迷路とN-パズルである．

迷路は入口(0,0)，出口(119,119)とする120×120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．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，上下左右に移動する各オペレータのコストは1とした．

またN-パズルとして48-パズルを用いた．これは7×7の盤面に1から48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，番号が正しく並ぶ目標状態へとタイルをスライドさせてゆく問題である．オペレータには空白の位置を上下左右のそれぞれに移動させる4つのものがあり，それぞれのコスト1とする．各状態の評価値としては各タイルの正しい位置とのマンハッタン距離の総和を用いた．

シミュレーション実験には障害物配置の異なる迷路と初期配置の異なる48-パズルをそれぞれ100個用意した．これらの問題はランダムに生成させたものであるが，すべて解を持つものである．そしてコミットメントリストの長さnを変化させ，アルゴリズム毎に各100回試行した．なお，実験に使用する計算機には利用可能な記憶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，探索は失敗として中断している．

表1 MSC-WA*の迷路における実験結果

アルゴリズム	成功率	ステップ数	解の長さ (最適解との比)
MSC-WA*(1)	100	1233.4	383.2 (1.24)
MSC-WA*(2)	100	1228.8	374.1 (1.21)
MSC-WA*(3)	100	1223.2	373.6 (1.21)
MSC-WA*(4)	100	1232.6	373.8 (1.21)
MSC-WA*(5)	100	1231.1	373.3 (1.21)
MSC-WA*(6)	100	1223.2	373.3 (1.20)
WA*	100	1229.9	373.5 (1.21)

表2 MSC-WA*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
MSC-WA* (1)	38	370374.0	9495.0
MSC-WA* (2)	97	190507.0	9470.7
MSC-WA* (3)	98	134848.0	10180.7
MSC-WA* (4)	100	120051.0	11138.1
MSC-WA* (5)	96	133266.0	11584.8
MSC-WA* (6)	99	157416.0	11622.1
WA*	10	189241.9	3151.0

表3 MSC-RTA*の迷路における実験結果

アルゴリズム	成功率	ステップ数	解の長さ (最適解との比)
RTA*	100	7413.1	363.5 (1.18)
MSC-RTA*(2)	100	7852.9	359.1 (1.16)
MSC-RTA*(3)	100	7918.5	358.5 (1.16)
MSC-RTA*(4)	100	8005.1	358.3 (1.16)
MSC-RTA*(5)	100	8108.2	358.1 (1.16)
MSC-RTA*(6)	100	8104.4	358.2 (1.16)

表4 MSC-RTA*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
MSC-RTA* (2)	100	155770.0	44569.0
MSC-RTA* (3)	100	93822.0	23426.0
MSC-RTA* (4)	100	120640.0	21231.0
MSC-RTA* (5)	100	181778.0	24243.0
MSC-RTA* (6)	100	210497.0	24512.0

MSC-WA*の実験結果を表1（迷路），表2（48-パズル），MSC-RTA*の実験結果を表3（迷路），表4（48-パズル）にそれぞれ示す．ここでアルゴリズム名における括弧内の数字はnである．成功率は記憶量の制限によることなく解を得た試行の割合である．ステップ数，解の長さはそれぞれ成功したものの平均である．

迷路ではMSC-WA*，MSC-RTA*ともにnに関わり

なく性能にほとんど変化がなかった．一方，48-パズルではともに劇的な性能改善が得られた．WA*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MSC-WA*は全ての問題で解を得ることができた．この実験からわかることは$n = 4$の時が最良の結果を得ており，それよりも大きくても小さくても性能は悪化する．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からである．

またRTA*ではわずか2%の問題しか解決できなかったのに対し，$n = 2$から$n = 6$ のMSC-RTA*では全て解を発見している．また$n = 3$としたときステップ数が最小となり，nはそれより大きくても小さくても性能が悪化する．

N-パズルに関してはサイズの異なる$N = 24, 35, 63$などの問題を用いた実験を行なったが同様の結果を得ており，この性能改善はN-パズルにおける一般的性質といえる．

さて，3，4節においてMSC-WA*，MSC-RTA*がともに解のコストに関してその保証を行うことができないことを指摘したが，実際に最適解とどの程度の違いがあるのかを示しておこう．迷路に関しては容易に最適解を求めることができ，表1，表3に示すようにMSC-WA*で1.21倍，MSC-RTA*で1.16倍ほど大きなコストの解が得られた．

48-パズルでは最適解を求めることは現在の計算機の能力では不可能である．そこで文献[Korf 88b]で示されている100個の15-パズル問題とその最適解を用いてその評価を行った．その結果，RTA*で最適解の20.16倍，$n = 6$の場合のMSC-RTA*で6.56倍であった．またWA*では3.90倍，$n = 6$の場合のMSC-WA*で8.00倍であった．

6. 考　　察

実験では多状態コミットメント探索はN-パズルにおいて劇的な性能改善を示すことを明らかにした．また性能を最適にするnの値が存在することも示した．一方で迷路においてはほとんど性能に違いが見られなかった．本節では以下の疑問に関して考察を加える．

- なぜ多状態コミットメントはN-パズルに対して効果的であったのに，迷路ではそうでなかったのか？
- N-パズルではなぜこれほどの性能改善がなされたのか?

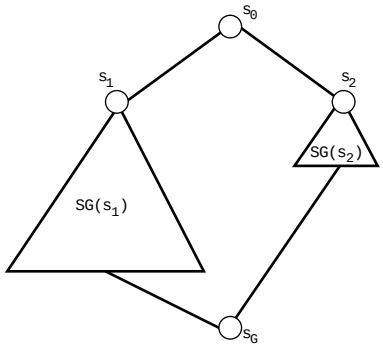


図5　探索グラフ

6・1　選択の誤りと回復

ここで多状態コミットメント探索がどのような問題に対して有効かを図5に示すグラフを用いて説明する．このグラフは状態s_0をもち，s_0は二つの子状態s_1とs_2を持つ．またそれぞれの子状態から部分グラフ$SG(s_1)$，$SG(s_2)$の探索が開始される．ここで$SG(s_1)$を経て目標状態s_Gに至るには多くの時間を要すが，$SG(s_2)$を経て目標状態に至るには短時間ですむと仮定する．またs_1とs_2に関する評価値は$h(s_1) < h(s_2)$の関係にあり，誤った選択を引き起こすとする．

このような問題において，単一状態コミットメント探索アルゴリズムは以下のように振舞う．探索はs_0から開始され，その子状態s_1とs_2が生成される．ここで誤った評価値のためにs_1が選択され，$SG(s_1)$が開始されるので，目標状態に至るまで多くの時間を費やすことになる．一方で多状態コミットメント探索アルゴリズムは以下のように振舞う．単一状態コミットメントと同様に探索はs_0から開始され，その子状態s_1とs_2が生成される．また誤った評価値のためにs_1が選択される．しかし多状態コミットメントではs_2も他の選択肢としてCOMMITMENTに保存される．そして$SG(s_1)$の探索中に$h(s_2)$の値がいずれCOMMITMENTの中で最小となり，アルゴリズムは$SG(s_1)$の探索を中断し，s_2から探索を再開し，目標状態に至る．このように多状態コミットメントは$SG(s_1)$の探索を中断できるならば，単一状態コミットメントよりも良い性能を示すことができる．すなわちs_2がCOMMITMENTに保存され，$SG(s_1)$の探索で多くの時間を浪費するまでに$h(s_2)$の値が最小になればよい．一般に，nが大きいほどs_2が保存される確率は増加する．

一方で，多状態コミットメントは以下の場合において副作用が生じることがある．今度は$SG(s_1)$のサイ

ズが $SG(s_2)$ のサイズより小さい場合を仮定する．単一状態コミットメントの場合は $SG(s_1)$ の探索を開始し，そのまま目標状態に至る．しかし多状態コミットメントでは $SG(s_1)$ の探索を中断し，$SG(s_2)$ の探索を開始することで，探索ステップを逆に増大させてしまう可能性がある．

さらに s_1 と s_2 以外にもその評価値が同じような選択肢が多数存在するような場合を想定してみよう．このとき n の値が大きいと，冗長な探索となるような選択肢の多くがCOMMITMENTに保存され，それらが選択されると，全体的な探索ステップ数を大きく増加させてしまうことが考えられる．これは n が大きすぎる場合のステップ数増加の原因であり，また全状態コミットメントの問題点も説明している．

以上の議論から多状態コミットメントは単一状態コミットメントに比較して，評価値が誤った選択を引き起こす可能性を含んでおり(条件1)，探索を途中で中断できる(条件2)ならより良い性能を示す．それではこれらの条件が評価実験で用いた迷路とN-パズルにおいて満たされているかどうかを調べてみることにしよう．

図6は迷路の一例である．迷路のサイズは 6×6 で，初期状態は(0,0)，目標状態は(5,5)である．また評価値として目標状態までのマンハッタン距離を用い，その値はそれぞれのマスの中に記入されている．この評価値は条件1を満たしている．すなわち初期状態(0,0)の子状態には s_1 と s_2 にそれぞれ該当する(1,0)と(0,1)が存在し，状態(0,1)を選択することが明らかに正しい選択である．ところが(0,1)と(1,0)の評価値はともに9であり，誤った選択を引き起こす可能性がある．

しかし，この評価値は条件2を満たしていない．すなわち(0,1)(= s_2)の評価値は図6にも示すように $SG(s_1)$ 内のいずれの状態の評価値を下回るものではない．したがって s_2 は n の値をいかに大きくしても $SG(s_1)$ の探索中にCOMMITMENTリストの中から選択されることはなく，このような性質をもつ迷路問題では多状態コミットメントはその性能を発揮することができない．

次に，図7は初期状態A，目標状態Gである15-パズルにおける状態空間グラフの一部を示したものである．この評価値も条件1を満たしている．すなわち初期状態Aの子状態B,C,D,Eの評価値がみな同じ値(10)であり，どの選択が望ましいかを区別することができない．この場合，次節で述べるようにBは誤った選択肢である．そしてこの評価値は条件2も満たしている．すなわち s_1 に該当するBの子状態の評価値は s_2 に該当するCのそれよりも大きい．したがって多状態コミットメントはいずれCOMMITMENTから s_2 を選択し，$SG(s_1)$ から s_2 に復帰するので，N-パズルでは効果的を発揮する．一方で，図7で示すように，N-パズルでは同じ評価値を持つ多数の状態が存在するので，大きな n をもつ多状態コミットメントや全状態コミットメントは探索範囲が広がりすぎ，探索に時間がかかったり，目標状態に到達する前に記憶領域を消費してしまう．

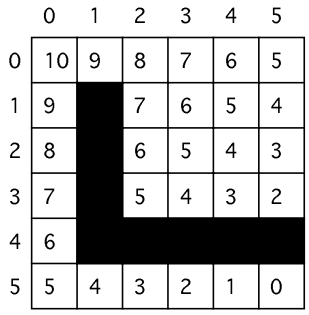


図6 迷路とその評価値

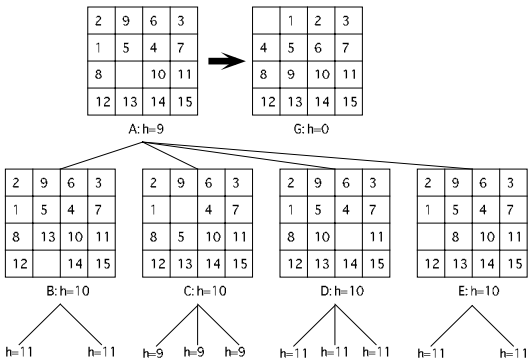


図7 15-パズルとその評価値の一部

6・2 直列化可能な副目標

多状態コミットメントはN-パズルにおいて劇的な性能改善を示した．図5に示されるように，多状態コミットメントと単一状態コミットメントの性能差は $SG(s_1)$ のサイズに依存する．つまり単一状態コミットメントが $SG(s_1)$ の探索をなかなか抜け出すことができないのに対し，多状態コミットメントは早い段階でそこから抜け出すことができるからである．

N-パズルにおいて誤った選択が大きな性能差を生むことは以下のような直列化可能な副目標[Korf 88a],[Newell 72]の概念を導入することにより説明される．直列化可能な副目標とは，目標が複数の副目標に分割され，かつある順番で副目標を順に達成する

場合, 以前の副目標を破壊せずに以降の副目標を達成することが可能であることをいう[Korf 88a]. 例えば, 15-パズルでは, 最初に一番下の並びのタイルを正しい位置に並べると, それ以降はこれらのタイルを動かすことなしに目標状態に到達可能である. さらに一番右の並びのタイルを正しい位置に並べると, これらのタイルを動かすことなしに目標状態に到達可能であり, 残りの問題は8-パズルと等価となる. 図8に示すように状態は副目標の達成の度合に応じてレベル付けをすることができる.

したがってN-パズルの探索では達成された副目標を壊さないように状態を選択してゆくことが望ましい. ところが異なるタイル間のマンハッタン距離の和に基づく評価値はこのような性質を反映していない. 図7の初期状態はすでにlevel2の副目標が達成されており, 状態Bの選択はそれを破壊することになる. しかしながら評価値は4つの子状態が全て同じ値であり, それを見分けることができない. そして一旦副目標を破壊するとその回復は容易でないので, 状態Bの選択は探索性能の悪化につながることが予想される. 一方, 多状態コミットメントの場合にはCOMMITMENTリストから他の選択肢に復帰することができるので, Bの選択は致命的ではない.

以上の議論を確認するために24-パズルにおける副目標の破壊の様子を表5に示す. 多状態コミットメントアルゴリズムにおける副目標レベルをCOMMITMENTリストにおける最大のものと定義する. 厳密には, 時刻Tにおけるアルゴリズムの副目標レベルSL^Tは

$$SL^T = max_{s \in \mathrm{COMMITMENT}^T} SL(s)$$

で定義される. ここで, $\mathrm{COMMITMENT}^T$は時刻TにおけるCOMMITMENTリスト, $SL(s)$は状態sの副目標レベルを表している. $n > 1$の場合, このSL^Tは時刻Tにおいて展開される状態の副目標レベルとは必ずしも等しくないことに注意していただきたい. アルゴリズムの副目標レベルが減少することはCOMMITMENTリストによる回復が無効になるほど致命的な選択誤りを行ったことを意味している. 表5に示した「副目標レベルを減少させる平均ステップ数」は探索ステップ数を副目標レベルを減少させた回数で割ることで計算され, アルゴリズムが致命的な選択誤りを行う頻度を表している.

表5からわかることは, nが大きければ副目標を破壊する頻度が少なくなるということである. すなわちアルゴリズムが一時的に副目標を破壊するような状態を選択したとしても, COMMITMENTリストに保存されている状態の中から, いずれそうでない選択肢に回復することができる. これはnを大きくすればするほどその効果が発揮されることを示している. しかしながら表2と表4が示すようにnを大きくしすぎると性能は悪化する. これは6.1節の最後にも述べたように探索範囲が広がりすぎることによるものであり, 副目標レベルを減少させにくくすることは, 逆にそれを増加させにくくもするという副作用が生じることを示している.

表5 24-パズルにおいて副目標レベルを減少させる平均ステップ数

アルゴリズム	ステップ数
RTA*	97.9
MSC-RTA*(2)	223.0
MSC-RTA*(3)	665.3
MSC-RTA*(4)	1327.3
MSC-RTA*(5)	1972.4
MSC-RTA*(6)	2719.3

7. 関 連 研 究

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) により, 並行して問題を解く, 協調探索と呼ばれる手法が提案されている. 協調探索においては複数のエージェントが存在するため, いずれかのエージェントが正しい選択をすれば問題を効率的に解くことができる. これまでに制約充足問題[Clearwater 91, Hogg 93]や状態空間探索問題[Knight 93, Kitamura 96]に対する適用例が報告されている. 協調探索の一つの問題点は, 重大な選択肢が連続して生じる場合, 探索の初期で誤った選択をしたエージェントはその後の探索に全く貢献しないことである. 文献[Yokoo 97]は, 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*) 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. 本研究は文献[Yokoo 97]に触発されたものであり, より単純なアルゴリズムで, 同様な性能向上が可能であることを示している. また, 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が, 本研究は単一エージェントによる探索であり, そのような問題点は解消されている. 例えば, [Yokoo 97]において, MRTA*アルゴリズムでは5エージェントが並行処理して48-パズルを解くのに平均35452.7ステップを要す

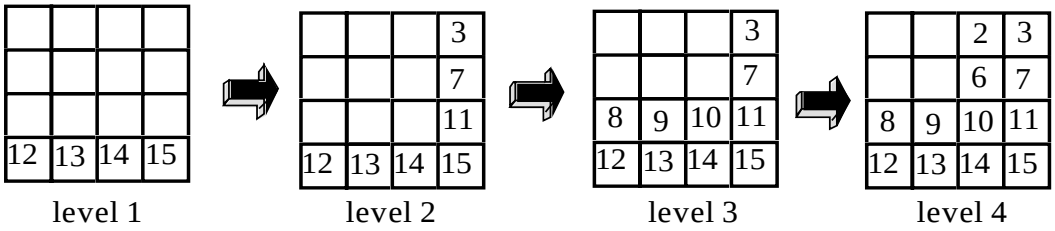


図8 副目標の達成

ることが報告されている.(全エージェントのステップ数の和は177263.5となる.) 一方$n = 3$とした場合のMSC-RTA*は93822.0ステップですんでいる.

探索の対象を絞る古典的な方法としてビーム探索[Bisiani 92]がある. 従来のビーム探索は探索範囲以外のものは完全に切り捨てているため, アルゴリズムの完全性が保証されないが, 本研究で提案する複数状態コミットメントの拡張においては, 元のアルゴリズム(RTA*, WA*)の完全性は保存されている.

探索の対象を限定するもう一つの方法として, Limited Discrepancy Search [Harvey 95, Korf 96]がある. これは, 深さ優先探索をベースとして, ヒューリスティック関数を用いて探索範囲を段階的に広げていく方法であり, 深さの上限があらかじめ分かっていない状態空間探索問題よりも, 探索木の深さの上限が自明である制約充足問題等に適している.

A*において, 有望でない状態をOPENから取り除き, 必要とされる記憶領域を削減するアルゴリズムとしてSMA*[Russel 92]がある. 本論文で提案するコミットメントの拡張は, このような記憶領域を制限する方法と排反ではなく, SMA* と組み合わせて用いることも可能である.

8. むすび

本研究では, 探索の範囲を制限する多状態コミットメントの概念をWA*とRTA*の二つの準最適解探索アルゴリズムに導入した. そして迷路, N-パズルを用いて実験評価を行いN-パズルでは劇的に性能が改善されることを示した. また本手法がどのような問題に対して効果的であるかに関する条件を示し, 直列化可能な副目標の視点からN-パズルがそれを満たしていることを示した.

今後の課題としてはさらに別の種類の問題に適用し, 本手法の有効性を示すことである. とくに与えられた問題に応じてnの値をどのように調整すべきかに関する指針を与えるべきであると考えている. さらに探索の過程で動的にnの値を調整するような手法も興味ある課題である. また本論文で提案したアルゴリズムでは解のコストに関する保証がないことを示したが, これを保証するような仕組みをアルゴリズムに組み込むことも今後の課題となっている.

謝辞

本研究に関して有益なコメントをいただいた京都大学大学院情報学研究科石田亨教授と査読者に感謝の意を表します.

◇参考文献◇

[Bisiani 92] Bisiani, R.: Beam search, In Shapiro, S. C., ed., *Encyclopedia of Artificial Intelligence*, pp.1467–1468, Wiley-Interscience Publication (1992).

[Clearwater 91] Clearwater, S. H., Huberman, B. A., and Hogg, T.: Cooperative solution of constraint satisfaction problems, *Science*, Vol.254, pp.1181–1183 (1991).

[Harvey 95] Harvey, W. D. and Ginsberg, M. L.: Limited discrepancy search, In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp.607–613 (1995).

[Hogg 93] Hogg, T. and Williams, C. P.: Solving the really hard problems with cooperative search, In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp.231–236 (1993).

[Kitamura 96] 北村, 寺西, 辰巳: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, 人工知能学会誌, Vol.11, No.3, pp.470–477 (1996).

[Knight 93] Knight, K.: Are many reactive agents better than a few deliberative ones?, In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp.432–437 (1993).

[Korf 88a] Korf, R. E.: Search in AI: A Survey of Recent Results, In Shrobe, H. E., ed., *Exploring Artificial Intelligence* , Morgan-Kaufmann (1988).

[Korf 88b] Korf, R. E.: Optimal Path-Finding Algorithms, In Kanal, L. and Kumar, V. eds., *Search in Artificial Intelligence* , Springer-Verlag (1988).

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

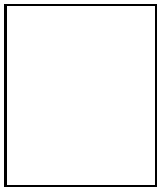
[Korf 93] Korf, R. E.: Linear-space best-first search, *Artificial Intelligence*, Vol.62, No.1, pp.41-78 (1993).

[Korf 96] Korf, R. E.: Improved limited discrepancy search, In *Proceedings of the Thirteenth National Con-*

ference on Artificial Intelligence, pp.286–291 (1996).
[Newell 72] Newell, A. and Simon, H. A.: *Human Problem Solving*, Prentice-Hall (1972).
[Pearl 84] Pearl, S.: *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley Publishing Company (1984).
[Russel 92] Russel, S.: Efficient memory-bounded search method, In *Proceedings of the Tenth European Conference on Artificial Intelligence*, pp.1–5 (1992).
[Shirai 82] 白井，辻井：人工知能，岩波書店 (1982).
[Yokoo 97] 横尾，北村：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, pp.47-55 (1997).

〔担当編集委員：××○○，査読者：○○××〕

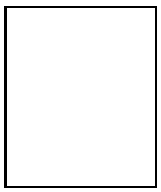
著 者 紹 介



北村 泰彦(正会員)

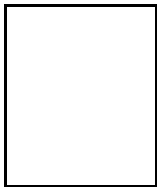
1983年大阪大学基礎工学部情報工学科卒業．1988年同大学院博士課程修了．工学博士．同年，大阪市立大学工学部電気工学科助手．現在，同情報工学科助教授．分散人工知能，ヒューリスティック探索，WWW情報統合の研究に従事．IEEE, AAAI, ACM, 電子情報通信学会，情報処理学会，ソフトウェア科学会等の会員．

〈kitamura@info.eng.osaka-cu.ac.jp〉



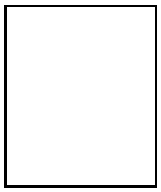
横尾 真(正会員)

前掲（Vol.14, No.4, p.?）参照．



宮地 智久

1997年大阪市立大学工学部情報工学科卒業．1999年同大学院修士課程修了．同年，住友電気工業（株）入社．在学中，ヒューリスティック探索の研究に従事．電子情報通信学会会員．



辰巳 昭治(正会員)

1970年大阪大学工学部通信工学科卒業．1992年同大学院修士課程修了．同年川崎重工業入社．1978年大阪大学大学院博士課程修了．工学博士．豊橋技科大学を経て，現在大阪市立大学工学部情報工学科教授．統計的パターン認識，意思決定問題，画像処理用並列プロセッサの開発，VLSI向き相互結合網の構成法などの研究に従事．電子情報通信学会，情報処理学会，ソフトウェア科学会，IEEE等の会員．

〈tatsumi@info.eng.osaka-cu.ac.jp〉