1 はじめに

準最適解を求めるヒューリスティック探索の高速化手法の一つである多状態コミットメント探索(Multi-State Commitment Search:MSCS)は，展開する状態を1つの状態に絞り込んでしまう1-状態コミットメント探索や，全く絞り込まない全状態コミットメント探索よりも効率的な探索を行うことが示されている[6] [1]．しかし，問題のどのような性質がMSCSの性能に影響を与えているのかは十分に説明されていなかった．

本稿では，[6]で提案されている多状態コミットメント実時間A* (Multi-State Commitment Real-Time A*:MSC-RTA*)アルゴリズムと多状態コミットメント重み付きA* (Multi-State Commitment Weighted A*:MSC-WA*)アルゴリズムについて性能解析を行う．具体的には，Nパズル問題，ハノイの塔問題，迷路問題におけるMSCSの動作を解析することで，MSCSが有効な問題の性質を解明する．また，そのような性質を持つ人工モデルを作成し，シミュレーション実験によって検証を行う．

2 ヒューリスティック探索の定式化

解析の前に，まずヒューリスティック探索の定式化を行う．問題は四つ組$\langle S, O, s_0, G\rangle$で表され，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．

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．

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定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．

3 MSC-RTA*アルゴリズム

実時間A*(Real-Time A*:RTA*)アルゴリズム[3]は一定の先読み探索と移動を交互に行うアルゴリズムである．先読みの深さで解の質と探索時間のいずれを重視するか選択でき，以下では探索時間を重視するため，先読みの深さを0としている．

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

図1. MSC-RTA*アルゴリズム

RTA*は状態を展開して隣接する状態を生成し，次に展開する状態を隣接する状態の中で最小の$f(s,s') = c(s,s') + h(s')$を持つものに決定する．探索は目標状態を生成したとき終了する．RTA*はクローズドリストを用いないが，次の状態を展開する前に$h(s)$を隣接する状態の中で二番目に小さい$f(s,s'')$に更新することでアルゴリズムの完全性を保証している．RTA*は隣接する状態の中で最小の評価値を持つ状態のみ保持するため，1-状態コミットメント探索と見なすことができる．

MSC-RTA*とRTA*の違いは，コミットメントリストを用いるかどうかである．MSC-RTA*において隣接する状態は一旦コミットメントリストに追加され，次に展開する状態はコミットメントリストから選択される．MSC-RTA*はパラメータとしてコミットメントリストの大きさの上限nを持ち，コミットメントリストがnよりも大きくなると，評価値の大きい状態から取り除かれる．このようにしてMSC-RTA*は次に展開する状態を1個ではなく，n個に絞り込む．$n=1$のMSC-RTA*はRTA*と等価である．MSC-RTA*のアルゴリズムを図1に示す．

いくつかの問題上でMSC-RTA*とRTA*の性能を比較した結果を以下に示す．まず，15パズル問題においてシミュレーション実験を行った．ヒューリスティックな評価関数に各タイルの目標状態における位置とのマンハッタン距離の和を用い，ランダムに生成した100個の問題で各100回，合計10,000回試行した．解を得るまでに状態を展開した回数について平均値を取ると，RTA*は2647.1回展開したが，$n=2$としたMSC-RTA*は1380.1回しか展開しなかった．

次に，円盤の数を7としたハノイの塔問題においてMSC-RTA*とRTA*が状態を展開した回数を調べた．初

[1]展開する状態をn個に絞り込むため，[6]ではn-状態コミットメント探索と呼ばれている．

期状態を全ての円盤がペグ 1 にある状態とし，全ての円盤をペグ 3 に移動させることで解決されるものとして，10,000 回試行した．なお，ヒューリスティックな評価関数には次式を用いた．

$$h(s) = \max(2m_1 - 1, 0) + \max(2m_2 - 1, 0) + 2m_3$$

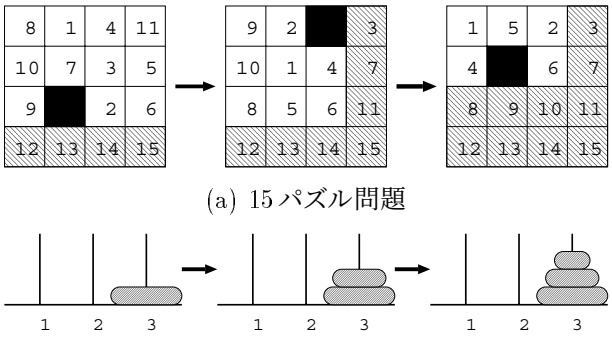
m_1，m_2 はペグ 1，ペグ 2 の円盤数，m_3 はペグ 3 にあるが正しくは置かれていない円盤数である．このとき，RTA* は 2150.9 回，状態を展開したが，$n = 15$ とした MSC-RTA* は 980.8 回しか状態を展開しなかった．

最後に，大きさを 120×120 とした迷路問題でシミュレーション実験を行った．ヒューリスティックな評価関数にはゴールとのマンハッタン距離を用い，障害物の割合を 40% としてランダムに生成した 100 個の問題で各 100 回，合計 10,000 回試行した．RTA* は 7413.1 回しか状態を展開しなかったにもかかわらず，$n = 2$ とした MSC-RTA* は 7852.9 回も状態を展開しており，n を大きくする程，より多く状態を展開した．

以上のように，MSC-RTA* は 15 パズル問題では $n = 2$ としたとき RTA* の約半分の展開数で，ハノイの塔問題でも $n = 15$ としたとき RTA* の半分以下の展開数で解を得ることができたが，迷路問題では RTA* よりも多くの展開数を必要とした ($n > 1$)．このように問題によって MSC-RTA* の効果が異なるのはなぜか？それぞれの問題の状態空間を調べ，MSC-RTA* と RTA* の動作を比較した結果，原因は副目標の保持と再展開の回数にあると予想される．

3.1 副目標の保持

目標が複数の副目標に分割でき，かつ副目標を順に達成する際に以前の副目標を保持したままで以降の副目標を達成できるとき，その問題は直列化可能な副目標を持つという．15 パズル問題は図 2(a) に示されているように，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残りのタイルを正しい位置に並べることができる．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残りのタイルを正しい位置に並べられ，残りの問題は 8 パズル問題と等価になる．以上より，15 パズル問題は直列化可能な副目標を持っている．そして，ハノイの塔問題も図 2(b) のように，一度ペグ 3 に正しく置かれた円盤を動かすことなく，他の円盤をペグ 3 に置くことができ，直列化可能な副目標を持つと考えられる．副目標は破壊されることもあり，もう一度達成するには余分な状態を展開しなければならないため，副目標を保持できるかが展開数を大きく左右する．



(a) 15 パズル問題

(b) ハノイの塔問題

図 2. 直列化可能な副目標

RTA* は展開する状態が 1 つに絞り込まれているため，一度副目標を破壊するような状態を選択するだけで副目標を保持できなくなる．例えば，図 2(a) 中央のようにタイルが配置されているとき，3 番のタイルを左に動かすと副目標は破壊されるが，2 番や 4 番のタイルを動かしても副目標は保持される．もし 3 番のタイルを左に動かしたならば，RTA* は 3 番のタイルを動かした配置だけを保持するため，副目標は保持されない．しかし，MSC-RTA* では 2 番や 4 番のタイルを動かした配置がコミットメントリストに残っている限り，いずれそれらの配置から探索を再開することができるため，副目標は保持される．

15 パズル問題上で 10,000 回試行した際に状態を展開した回数と副目標を一度も壊さずに解くことができた問題の数の平均値を表 1 に示す．15 パズル問題において，$n = 2$ とした MSC-RTA* は RTA* の 4 倍も副目標を保持しやすいことが分かる．なお，$n = 6$ とした MSC-RTA* は $n = 2$ としたものよりも副目標を保持しやすいにもかかわらず，状態を展開した回数は多くなっているが，この理由としては MSC-RTA* が RTA* よりも展開する状態の絞り込みが緩いため，より多くの空間を探索しなければならなかったことがあげられる．副目標を破壊することなく問題を解くこと場合の結果にのみ注目すると，探索空間の増大による展開数の増加が明確になる．副目標を破壊することなく問題を解くことができた際に，状態を展開した回数と，展開した状態の個数，すなわち探索した空間の大きさの平均値を表 2 に示す．

また，ハノイの塔問題を解く際に副目標を保持できた問題の数と状態を展開した回数を表 3 に示す．RTA* では全体の約 4 分の 3 の問題でしか，副目標を保持できなかったが，$n = 15$ とした MSC-RTA* ではほぼ全ての問題で副目標を保持できている．n を大きくすると副目標を完全に保持できる反面，展開数が増加するのは 15 パズルと同じく，絞り込みを緩くすると探索空間が増大す

表1. 15パズル問題における副目標の保持

アルゴリズム	展開数	副目標を保持できた問題数
RTA*	2647.1	216
MSC-RTA*($n=2$)	1380.1	984
MSC-RTA*($n=6$)	1600.3	4986

表2. 副目標を保持できた場合の展開数と空間量

アルゴリズム	展開数	空間量
RTA*	641.5	579.7
MSC-RTA*($n=2$)	646.0	579.6
MSC-RTA*($n=6$)	1127.5	1009.1

るためである．

以上より，直列化可能な副目標を持つ問題では，MSC-RTA*がRTA*よりも副目標を保持しやすいため，結果としてRTA*より少ない展開数で解が得られると考えられる．

3.2 再展開数の減少

MSC-RTA*やRTA*はクローズドリストを用いないため，状態の再展開が起こり得るアルゴリズムである．従って，同じ領域を探索しても再展開の回数が少なければ，結果として少ない展開数で解が得られる．MSC-RTA*が状態の再展開を抑制する仕組みを説明するための例として，隣接する状態が全て展開済みである状況を考える(図3)．RTA*は隣接する状態のみから次に展開する状態を選択するため，必ず状態の再展開が起こる．しかし，MSC-RTA*では隣接する状態よりも小さい評価値を持つ未展開の状態がコミットメントリストに存在すれば，その状態が次に展開されるため再展開は生じない．

ハノイの塔問題において，未だペグ3に正しく置かれていない円盤の枚数が等しい状態のほとんどで$h(s)$は等しくなる．未だペグ3に正しく置かれていない円盤の枚数をMとし，ペグ1，ペグ2に置かれた円盤の枚数をそれぞれm_1，m_2，ペグ3にあるが，正しくは置かれていない円盤の枚数をm_3とすると，図4のように状態は$m_2 = 0$の場合と$m_2 \neq 0$の場合に分けることができる．なお，m_1とm_2を入れ換えても同様である．ペグ3に正しく置かれた円盤の枚数が変化しない限り，$h(s)$は$2M-1$と$2M-2$の2値しか取らず，$m_2 \neq 0$となる状態の数$(2 \cdot 3^{M-1} - 2^{M-1})$は$m_2 = 0$となる状態の数$(2^{M-1})$より遥かに多いため，$M$枚の円盤がペグ3に正しく置かれていない状態の大部分で$h(s) = 2M-2$であ

表3. ハノイの塔問題における副目標の保持

アルゴリズム	展開数	副目標を保持できた問題数
RTA*	2150.9	7456
MSC-RTA*($n=15$)	980.8	9982
MSC-RTA*($n=30$)	1022.8	10000

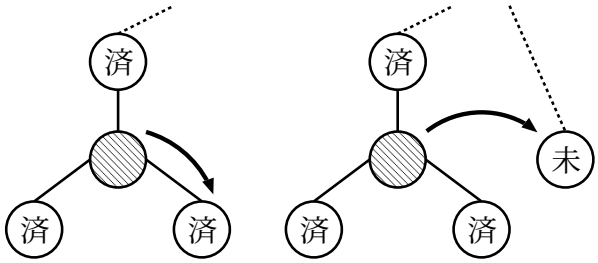


図3. MSC-RTA*による再展開の回避

る．このように同じ評価値を持つ状態が密集した領域では，一度でも展開した状態の評価値は必ず未展開の状態よりも大きくなるため，MSC-RTA*の再展開数はRTA*よりも少なくなる．表4に円盤の数を7としたハノイの塔問題において，MSC-RTA*とRTA*が状態を再展開した数を示す．MSC-RTA*がRTA*の約10分の1しか再展開をしないことが分かる．

また，ヒューリスティックな評価関数にゴールへのマンハッタン距離を用いた迷路問題ではMSC-RTA*はRTA*よりも再展開数が多くなる(表5)．これは図5のように袋小路の奥ほど評価値が小さくなる傾向があり，MSC-RTA*では，袋小路の状態の評価値がコミットメントリスト内で最小でなくなるまで，袋小路の状態を何度も再展開するためである．

以上より，同じ評価値を持つ状態が密集している問題では，MSC-RTA*がRTA*よりも状態を再展開しにくいため，結果としてMSC-RTA*がRTA*よりも少ない回数しか状態を展開せずに解が得られると考えられる．

4 MSC-WA*アルゴリズム

A*アルゴリズム[5]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推定値$h(s)$の和を用いるが，重み付きA*(Weighted A*:WA*)アルゴリズム[4]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$h(s)$に対する重みであり，Wによって解の質と探索時間

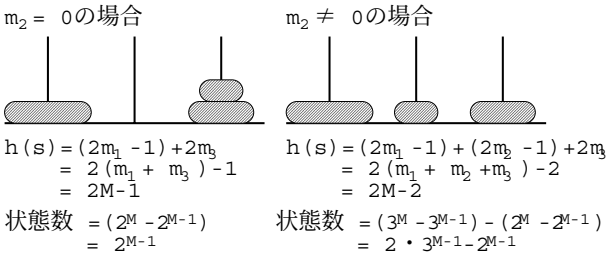


図4. ハノイの塔問題における$h(s)$のパターンとそれぞれの状態数

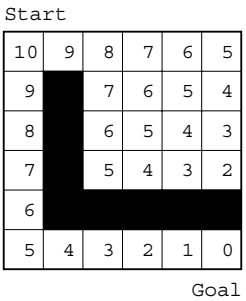


図5. 迷路問題

表4. ハノイの塔問題における再展開数

アルゴリズム	展開数	再展開数
RTA*	2150.9	1136.3
MSC-RTA*($n=15$)	980.8	115.2

表5. 迷路問題における再展開数

アルゴリズム	展開数	再展開数
RTA*	7413.1	6163.0
MSC-RTA*($n=2$)	7852.9	6606.5
MSC-RTA*($n=6$)	8104.4	6863.9

のいずれを重視するかを決定できる．以下では，探索時間を重視するため，$W=1$の場合を想定し，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，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*は未展開の状態を全てオープンリストに保持しており，展開する状態が全く絞り込まれていないため，全状態コミットメント探索と考えることができる．

MSC-WA*では，WA*のオープンリストの一部をコミットメントリストとし，WA*がオープンリストに対して行う処理をコミットメントリストに対して行う．すなわち，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追加され，次に展開する状態はコミットメントリストから選択される．MSC-WA*はパラメータとしてコミットメントリストの大きさの上限nを持ち，コミットメントリストがnよりも大きくなると，評価値の大きい状態からオープンリストへ移し変えられる．また，コミットメントリストがnより小さければ，オープンリスト内で評価値の小さい状態からコミットメントリストへ移し変える．このようにしてMSC-WA*は次に展開する状態をn個に絞り込む．$n=\infty$のMSC-WA*はWA*と等価である．MSC-WA*のアルゴリズムを図6に示す．

MSC-WA*とWA*を用いてMSC-RTA*と同様の条件でシミュレーション実験を行った．まず，15パズル問題において，状態をWA*は1889.0回展開したが，$n=4$としたMSC-WA*は1374.0回しか状態を展開しなかった．そして，ハノイの塔問題ではWA*は状態を838.4回展開し，$n=2$としたMSC-WA*は796.4回，状態を展開した．最後に，迷路問題では，WA*は状態を1229.9回展開し，$n=2$としたMSC-WA*は1223.7回，状態を展開した．わずかながらMSC-WA*の方が少ないが，これはほとんど誤差の範疇である．

以上のように，MSC-WA*は15パズル問題ではWA*の約3分の2しか状態を展開せずに解を得ることができたが，ハノイの塔問題や迷路問題ではWA*と大差ない性能しか示すことができなかった．この原因がどこにあるかを調べた結果，それぞれの問題における刈り込みの有効性にあると予想した．

4.1 刈り込みの有効性

MSC-WA*がWA*のオープンリストをコミットメントリストとオープンリストに分割し，コミットメントリスト内の状態から探索を勧めて行くことは一時的な刈り込みと考えることができる．刈り込みの効果は余分な空間を探索しないことにあるが，刈り込んだ場合と刈り込まなかった場合で同じ空間を探索していれば，刈り込みは無意味になってしまう．

15パズル問題，ハノイの塔問題，迷路問題における状態を展開した回数と，解が得られた際，コミットメントリスト，オープンリスト，クローズドリストに保存されていた状態の和，すなわち探索した空間の大きさを表6

```
MSC-WA*(⟨S, O, s0, G⟩)
1:  s ← s0;
2:  generate successors(s);
3:  if successors(s) ∩ G ≠ φ then return success;
4:  add(commit_list, successors(s));
5:  add(closed_list, s);
6:  while length(commit_list) > n do
7:      s ← get_max(commit_list);
8:      add(open_list, s);
9:  while length(commit_list) < n and
10:         (open_list ≠ φ) do
11:     s ← get_min(open_list);
12:     add(commit_list, s);
13: if (commit_list = φ) and (open_list = φ) then
14:     return failure;
15: s ← get_min(commit_list);
16: goto 2;
```

図6. MSC-WA*アルゴリズム

に示す．15パズル問題において$n = 4$としたMSC-WA*はWA*よりも少ない空間しか探索していないが，ハノイの塔問題や迷路問題ではWA*とほぼ同じ大きさの空間を探索している．

刈り込みの有効性を示す指標として，$n = 1$としたMSC-WA*を用いて，状態を展開した際，新しい状態を生成しないことがどれくらいあるかを測定した(表7)．すなわち，極端に刈り込まれた状況で状態を展開した回数と比して新しい状態を生成しないことが多ければ，オープンリストからたびたび補充を受けることになり，刈り込みの効果が小さいと考えられる．15パズル問題ではほぼ毎回新しい状態を生成しているが，ハノイの塔問題や迷路問題では状態を展開してもおよそ4回に1回は生成済みの状態しか生成できず，オープンリストから補充を受けており，刈り込みがあまり有効でないことを示している．なお，15パズル問題で$n = 1$としたMSC-WA*よりも$n = 4$としたものの方が良い性能を示すのは，刈り込みを厳しくしすぎると副目標の保持が困難になるためである．

以上より，状態を展開するたびに未生成の状態の状態を生成するような問題では，刈り込みの効果によってMSC-WA*がWA*よりも少ない展開数で解を得られると考えられる．

表6. それぞれの問題における展開数と空間量

アルゴリズム	展開数	空間量
MSC-WA*($n = 4$)	1374.0	2886.6
WA*	1889.0	3905.6

(a) 15パズル問題

アルゴリズム	展開数	空間量
MSC-WA*($n = 2$)	796.4	876.3
WA*	838.4	891.3

(b) ハノイの塔問題

アルゴリズム	展開数	空間量
MSC-WA*($n = 2$)	1223.7	1427.4
WA*	1229.9	1429.4

(c) 迷路問題

表7. 各問題における新しい状態を生成しない回数

問題	展開数	新規生成なしの回数
15パズル問題	3940.7	9.2
ハノイの塔問題	900.7	219.8
迷路問題	1233.4	294.3

5 人工モデル

MSC-RTA*の性能が副目標の保持と再展開数の減少に依存し，MSC-WA*の性能が刈り込みの有効性に依存していることを確かめるため，3つの人工モデルを作成し，シミュレーション実験によって検証を行った．

まず，MSC-RTA*がRTA*よりも副目標を保持しやすいことを示すため，直列化可能な副目標を持つ人工モデルとして，中央の障害物によって$h(s)$の初期値が異なる2つの領域に分割された迷路問題を作成した(図7(a))．左上が初期状態であり，右下が目標状態である．障害物の左側の領域から右側の領域に移動することで副目標は達成されるが，領域をつなぐ通路は2つしかない上に，領域間の移動の際，$h(s)$の更新によって通路の$h(s)$は周囲よりも大きくなるため，右側の領域から左側の領域に戻ってしまうと，通路周辺の$h(s)$が通路の$h(s)$よりも大きくなるまで，もう一度右側の領域に移動できない．このため，副目標を破壊した場合，もう一度達成するには余分に状態を展開する必要がある．大きさが30×30の人工モデルで10,000回試行した際に副目標を保持できなかった問題の数と展開数の平均値を表8(a)に示す．MSC-RTA*はRTA*の16分の1の問題でしか副目標の保持に失敗しておらず，結果としてRTA*より少ない展開数で解を得

ている．表8(b)はこの人工モデルおいて探索中に副目標を保持できた場合と保持できなかった場合の展開数を示しており，副目標を保持できなかった場合は保持できた場合の2倍以上も状態を展開しなければならないことが分かる．

この人工モデルでは，障害物の左側の領域では$h(s) = 2$だが，右側の領域では$h(s) = 1(< 2)$であるため，15パズル問題やハノイの塔問題と同様に副目標を保持している状態の方が副目標を保持していない状態より$h(s)$が小さくなっている．このため，たとえ副目標を破壊するような状態を選択したとしても，副目標を保持している状態がコミットメントリストに残りやすく，MSC-RTA*ではRTA*よりも副目標の保持が容易である．

MSC-RTA*においてnを大きくするほど副目標を保持しやすくなる条件に，副目標を保持している状態の$h(s)$が保持していない状態の$h(s)$よりも小さくなることが挙げられる．この人工モデルのように条件を満たす問題では，副目標を破壊されていない状態がコミットメントリストに残りやすい．ハノイの塔問題では副目標を保持している状態の$h(s)$は保持していない状態よりも必ず小さくなり$(\max\{2M-1, 2M-2\} < \min\{2(M+1)-1, 2(M+1)-2\})$，15パズル問題でも副目標を達成するほど，残りのタイルが減るため$h(s)$の最大値は小さくなる．

また，副目標の保持の容易さには副目標を達成する経路の数も関係しており，経路の数が多いと副目標は達成しやすいが，保持は困難になり，この人工モデルのように経路の数が少ないと副目標を達成しにくい反面，保持は容易になる傾向がある．15パズル問題では副目標を達成するタイルの並びには何通りものパターンがあるため，副目標の達成方法が多く，ハノイの塔問題では最大の円盤をペグ1からペグ3へ，またはペグ2からペグ3へ置くというわずか2通りの方法しか副目標を達成できないため，副目標の達成方法は極端に少ないといえる．このことは15パズル問題と比べてハノイの塔問題で副目標を保持しやすいことの原因となっている．

次に，MSC-RTA*がRTA*よりも状態を再展開しにくいことを示すための人工モデルとして，障害物を含まない迷路問題を作成し，各状態の$h(s)$の初期値を全て1とした(図7(b))．この人工モデルではヒューリスティックが目標状態に関する情報を全く示さないため，ランダムな選択を繰り返して，隣接する状態が全て展開済みの状況を作り，状態を再展開しやすい．大きさが30×30の人工モデルで10,000回試行した結果を表9に示す．MSC-RTA*はRTA*の4分の1しか再展開を行わず，RTA*より少ない展開数で解を得ることができた．これは未展開の状態は全て$h(s) = 1$だが，一度でも展開した状態では$h(s) > 1$となるため，MSC-RTA*ではコミットメントリストに未展開の状態がある限り再展開は起こらないからである．

表8. 副目標を持つモデルにおける実験結果

アルゴリズム	副目標を保持できなかった問題の数	展開数
RTA*	3353	1075.7
MSC-RTA*($n=2$)	728	687.2
MSC-RTA*($n=3$)	398	628.4
MSC-RTA*($n=4$)	297	610.8
MSC-RTA*($n=5$)	202	597.4

(a) 副目標を保持できなかった問題の数と展開数

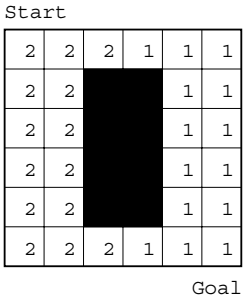
アルゴリズム	保持できた場合	保持できなかった場合
RTA*	708.7	1803.2
MSC-RAT*($n=5$)	573.2	1770.4

(b) 副目標保持の可否による展開数の相違

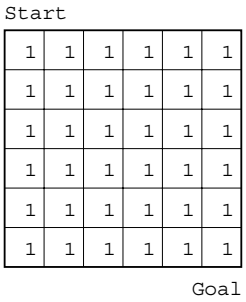
表9. 初期値が全て等しい人工モデルにおける実験結果

アルゴリズム	再展開数	展開数
RTA*	453.5	1161.6
MSC-RTA*($n=2$)	182.5	775.2
MSC-RTA*($n=3$)	145.4	728.7
MSC-RTA*($n=4$)	127.0	713.7
MSC-RTA*($n=5$)	115.8	708.1

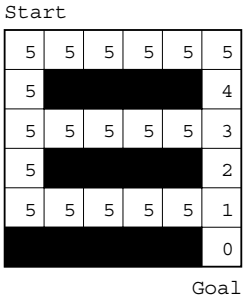
最後に，図7(c)に示される刈り込みが有効な人工モデルを用いてMSC-WA*がWA*より小さな空間だけを探索することを示す．この人工モデルにおいて，一番右の列の状態だけは$h(s)$がゴールまでのコストを正確に示しているが，他の状態の$h(s)$は全て等しいため，一番右の列へ到達するまでどの経路も同等に望ましくみえる．どの経路を選んでも結局は同じ展開数で解が得られるのだが，途中までは全ての状態が同じ評価値を持つため，刈り込みを行わなければ，余分な空間を探索してしまう．大きさが30×30の人工モデルで10,000回試行した結果を表10に示す．WA*は全ての経路を探索しようとするが，$n = 1$としたMSC-WA*はいずれか1つの経路のみ探索するため，結果としてMSC-WA*はWA*の約4分の1しか状態を展開せずに解を得ることができた．



(a) 副目標の達成と対応した$h(s)$を持つ人工モデル



(b) $h(s)$の初期値が全て等しい人工モデル



(c) 刈り込みが有効な人工モデル

図7. 人工モデル

6 関連研究

MSC-RTA*はマルチエージェント実時間A*アルゴリズム[2]に確率的再配置を導入した探索アルゴリズム[7]からヒントを得て考案された．そのため，これらもMSC-RTA*と同様にRTA*より副目標を保持しやすくなっており，直列化可能な副目標を持つ問題では非常に有効である．しかし，複数のエージェントが同じ状態を展開することがあるため，MSC-RTA*より効率が悪い．

MSC-WA*と良く似た刈り込みを行う探索アルゴリズムにビーム探索[1]があり，MSC-WA*が有効な性質を持つ問題ではビーム探索も同じく有効である．しかし，MSC-WA*は完全性が保証されているが，ビーム探索は完全性が保証されていないため，MSC-WA*では解くことができても，ビーム探索では解けない問題が存在する．

表10. 刈り込みが有効な人工モデルにおける実験結果

アルゴリズム	展開数	空間量
MSC-WA*($n=1$)	58.0	72.5
MSC-WA*($n=2$)	83.9	99.5
MSC-WA*($n=3$)	110.4	127.3
MSC-WA*($n=4$)	134.0	152.5
MSC-WA*($n=5$)	156.4	176.5
WA*	224.2	248.8

7 むすび

15パズル問題，ハノイの塔問題，迷路問題におけるMSCSの動作を調べることにより，MSCSが有効な問題の性質を明らかにした．人工モデル上のシミュレーション実験によって，MSC-RTA*がRTA*より副目標を保持しやすく，再展開の回数も少ないため，RTA*より少ない展開数で解が得られることを示した．また，刈り込みが有効な人工モデルでWSC-WA*がWA*より少ない展開数で解を得ることを示した．

参考文献

[1] Bisiani, R. Beam search. In Shapiro, S. C., ed., *Encyclopedia of Artificial Intelligence*. New York: Wiley-Interscience Publication. 1467–1468. 1992.

[2] K.Knight. Are Many Reactive Agents Better Than a Few Deliberative Ones?. *Proc. of IJCAI-93*, 432–437. 1993.

[3] Korf, R. E. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211. 1990.

[4] Korf, R. E. Liner-space best-first search. *Artificial Intelligence*, 62(1):41–78. 1993.

[5] J. Pearl. Heuristics. Addison Wesley. 1984.

[6] 宮地 智久, 北村 泰彦, 横尾 真, 辰巳 昭治. 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. 信学技報, AI97-58. 1998.

[7] 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, 14(4):47–55. 1997.