ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.2, February 2009, pp.290−304
doi: 10.3724/SP.J.1001.2009.03152

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

部分可观察强规划中约减观察变量的研究*

周俊萍[1], 殷明浩[1,2,3+], 谷文祥[1], 孙吉贵[2,3]

[1](东北师范大学 计算机学院,吉林 长春 130117)

[2](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[3](吉林大学 教育部符号计算与知识工程重点实验室,吉林 长春 130012)

Research on Decreasing Observation Variables for Strong Planning under Partial Observation

ZHOU Jun-Ping[1], YIN Ming-Hao[1,2,3+], GU Wen-Xiang[1], SUN Ji-Gui[2,3]

[1](School of Computer Science, Northeast Normal University, Changchun 130117, China)

[2](School of Computer Science and Technology, Jilin University, Changchun 130012, China)

[3](Key Laboratory of Symbolic Computation and Knowledge Engineering for the Ministry of Education, Jilin University, Changchun 130012, China)

+ Corresponding author: E-mail: mhyin@nenu.edu.cn

Zhou JP, Yin MH, Gu WX, Sun JG. Research on decreasing observation variables for strong planning under partial observation. *Journal of Software*, 2009,20(2):290−304. http://www.jos.org.cn/1000-9825/3152.htm

Abstract: How to decrease the observation variables for strong planning under partial observation is explored. Beginning from a domain under no observation, add necessary observation variables gradually to get a minimal set of observation variables necessary. Two methods are presented to decrease observation variables. With the former, when any of the two distinct states of the domain can be distinguished by an observation variable, this algorithm can find a minimal set of observation variables necessary for the execution of a plan. With the latter, when there are states that can't be distinguished by only one observation variable, this algorithm can find a set of observation variables as small as possible which are necessary for the execution of a plan.

Key words: strong planning; planning under partial observation; strong planning under partial observation; decrease observation variables; nondeterministic planning

摘 要: 给出了一种约减观察变量方法——假设所有的状态变量都不是观察变量,在此基础上逐步增加必要的观察变量,从而最终得到一个必要的观察变量集合.在添加必要的观察变量过程中,该方法不要求得到所有变量的相关信息,从而具有更好的通用性.根据是否存在单个观察变量能够区分域中任意两个状态的问题,分别给出了两种约减观察变量方法:当存在一个观察变量可以区分规划域中任意两个状态时,算法可以得到一个最小的观察变量集合;当不存在这样一个观察变量时,算法可以得到一个尽可能小的观察变量集合,但不能保证该集合最小.

* Supported by the National Natural Science Foundation of China under Grant Nos.60496321, 60473042, 60573067, 60803102 (国家自然科学基金); the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No.20050183065 (高等学校博士学科点专项科研基金); the Science Foundation for Young Teachers of Northeast Normal University of China under Grant No.20070601 (东北师范大学青年基金)

Received 2007-03-07; Accepted 2007-09-04

关键词: 强规划;部分可观察规划;部分可观察强规划;约减观察变量;不确定规划
中图法分类号: TP18 **文献标识码:** A

在过去 10 年中,经典智能规划的研究取得了飞速的进展,相对于早期的智能规划系统,现代的智能规划系统无论在规划求解效率上还是在规划求解规模上都有指数级的提高.例如,在 2006 年举办的第 5 届国际智能规划比赛中,SGplan 等规划系统已经可以在较短时间内处理 10^{20} 以上的状态.然而,经典智能规划方法通常假设智能规划 agent 对于规划世界的知识是完全的,规划过程中动作的效果是确定的,动作被看作状态的瞬间转移而不是持续性的效果[1],这在很大程度上限制了智能规划的应用范围.因而强规划等非经典规划问题日益受到研究人员的重视,如 2004 年举办的第 4 届国际智能规划竞赛中首次设置了概率规划组的比赛,2006 年举办的第 5 届国际智能规划大赛则首次设置了一致性规划组竞赛.

强规划问题是指在初始状态和动作的执行效果都具有不确定因素的情况下,依然能够保证找到从初始状态到达目标状态的动作序列的规划问题.根据强规划执行过程中需要观察变量的程度,它可以分为完全可观察的强规划问题、部分可观察的强规划问题和完全不可观察的强规划问题即一致性规划问题.近年来,部分可观察的规划问题受到了研究人员的重视,并给出了多种求解方法.文献[2]在可能语义框架中将部分可观察的规划问题转换为搜索问题,并利用深度优先搜索方法进行求解.该方法在初始条件和动作的执行效果都不确定的条件下,可以生成一个条件规划,保证从初始状态到达目标状态.文献[3]处理了部分可观察下的强规划问题.它定义了一个通用的结构来模拟部分可观察规划域并且提出了一种在信念空间中基于与或搜索的规划算法.这种算法通过给出几种不同的搜索策略使其在生成规划时更加有效.文献[4]采用了交叉规划和执行的方法.它不像其他处理部分可观察规划方法——在庞大的与或图中进行搜索,而是采用另外一种策略——规划器在部分图中搜索,同时控制器执行部分规划并且监视域中的当前状态,因而它可以处理带有庞大状态空间的规划问题.文献[5]采用了逻辑框架方法.它给出了一种如何把动作的效果从状态层次提升到认知层次的方法,这种结构使它既可以采用前向搜索,也可以采用后向搜索.文献[6]采用了部分可观察马尔可夫决策过程.它给出了一种向量空间方法,该向量空间模型为信念状态近似值问题和该近似值如何影响决策质量提供了新的见解.

事实上,如果将强规划问题看作一个光谱,光谱的一端为这样一类强规划问题,在这类问题中,所有的状态变量都必须是观察变量,在规划求解的过程中需要利用状态变量的所有信息;光谱的另一端为一致性规划问题,在这类问题中,所有的状态变量都不是观察变量,在规划求解的过程中我们不需要任何观察依然可以求解规划.在光谱的中间存在着这样一类强规划问题,在求解过程中需要一些状态变量是可观察的,这样我们才能保证规划可以求解.在文献[2–6]中,所有的观察变量都是事先手工给定的.事实上,我们知道,设定观察变量的意义在于获得信息,而任何信息的获取都需要相应的花费.因而给定一个部分可观察强规划问题,由计算机自动确定一个最小(或尽可能小)的必要观察变量集合(即确定该问题在光谱中所处的位置),从而根据此变量集合求解该规划问题往往是非常必要的.中山大学的姜云飞教授等人在文献[7]中首次给出了一种确定最小观察变量集合的方法.该方法的基本思想是:给定一个强规划问题,首先假设所有状态变量为观察变量,在此基础上逐步删除不必要的观察变量,最终得到最小的必要观察变量集合.然而在很多情况下,我们无法对所有的状态变量进行观察,因而本文从另一个角度来处理该问题.初始时,我们允许所有的状态变量都不是观察变量,在此基础上逐步增加必要的观察变量,从而最终得到一个最小的必要观察变量集合.此外,文献[7]在求解过程中假设存在一个观察变量能够区分规划问题中的任意两个状态,但在现实生活中存在大量任何单个观察变量都不能区分两个状态的情况[3],因而本文分别从两种情况考虑该问题,并分别给出了求解方法:当存在一个观察变量,可以区分规划域中任意两个状态时,我们的算法得到一个最小的观察变量集合;当不存在这样一个观察变量时,我们得到一个尽可能小的观察变量集合,但是我们不能保证该集合最小.

1 相关概念

为了方便理解本文,我们在这部分给出与本文相关的一些定义.更详细的定义参见文献[2,3,8–13].

定义 1(不确定规划域). 不确定规划域 $D=\langle I,S,A,R\rangle$ 是一个四元组,其中, $I\subseteq S$ 是一个初始状态集合,S 是一个有限的状态集合,A 是一个有限的动作集合, $R:S\times A\to 2^S$ 是一个状态转换函数.

例 1:图 1 所示为一个简单的机器人导航问题,它包含 3×4 个房间,分别为 $s_0,s_1,...,s_{11}$.机器人可位于域中的任意一个房间,即有限的状态集合为 $S=\{s_0,s_1,...,s_{11}\}$;机器人可以向 4 个方向移动,即有限的动作集合为 $A=\{Gonorth,GoSouth,GoEast,GoWest\}$.一个动作只有当它的目标位置没有墙阻隔时,才是可用的,如机器人在状态 s_1 时,动作 *GoNorth* 是不可用的,即状态转移函数 $R(s_1,GoNorth)=\varnothing$;动作 *GoEast* 是可用的,即状态转移函数 $R(s_1,GoEast)=\{s_2\}$;而且机器人在状态 s_1 和 s_4 时,动作 *GoWest* 是不可用的.动作分为两种:确定的和不确定的,机器人在状态 s_0 和 s_3 时,动作 *GoEast* 是不确定的,即状态转移函数 $R(s_0,GoEast)=\{s_1,s_4\}$, $R(s_3,GoEast)=\{s_1,s_4,s_7\}$;其他所有动作都是确定的.机器人的初始状态可能是 s_0 或 s_3,即 $I=\{s_0,s_3\}$,目标状态是 s_6,即 $G=\{s_6\}$.

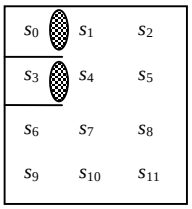


Fig.1 A simple robot navigation problem

图 1 一个简单的机器人导航问题

定义 2(不确定规划问题). 不确定规划问题 $P=\langle D,I,G\rangle$ 是一个三元组,其中,D 为不确定规划域,I 是初始状态集合,G 是目标状态集合,且 I,G 均不为空.

定义 3(部分可观察不确定规划域). 部分可观察不确定规划域 $\Sigma=\langle S,A,R,O\rangle$ 是一个四元组,其中 S 是一个有限的状态集合,A 是一个有限的动作集合,$R:S\times A\to 2^S$ 是一个转移函数,O 是一个有限的观察变量集合.

定义 4(观察函数). 令 $\Sigma=\langle S,A,R,O\rangle$ 是一个部分可观察不确定规划域,在 S 和 O 上的观察函数是 $X:S\times O\to\{T,F\}$.

定义 5(可用动作). 令 $Bs\subseteq S$ 是一个信念状态,动作 a 在 Bs 上是可用的,当且仅当 $Bs\neq\varnothing$ 且 a 可用于 Bs 中的任意一个状态.

我们所说的信念状态是指智能体通过选择一定的策略后可能所处的状态集合.例如,在机器人导航问题中,机器人在状态 s_3 执行动作 *GoEast* 后,智能体可能所处的状态是 s_1,s_4 和 s_7,那么,该信念状态为 $Bs=\{s_1,s_4,s_7\}$.

定义 6(动作的映像). 令 $Bs\subseteq S$ 是一个信念状态,动作 a 在 Bs 上是可用的.a 在 Bs 上的映像记作 $Image[a](Bs)$,定义如下: $Image[a](Bs)=\{s'\mid\exists s\in Bs$使$R(s,a)=\{s'\}\}$.

定义 7(一致性规划). 规划 π 是规划问题 $P=\langle D,I,G\rangle$ 的一个一致性规划,当且仅当以下条件成立:

(1) π 在 l 上是可用的;(2) $Image[\pi](I)\subseteq G$.

实质上,一致性规划也就是无传感器规划.这种规划在执行期间不需要任何传感器,它不管真实的初始状态和实际行动的结果是什么,都要找到在各种可能的环境中完成目标的规划[14−17].因而,可以说它是完全不可观察的强规划.

2 算法概述

在文献[7]的约减观察变量算法中,对于给定的一个强规划问题,该算法假设所有状态变量都为观察变量,在此基础上逐步删除不必要的观察变量,最终得到最小的必要观察变量集合.然而在很多情况下,我们无法对所有的状态变量进行观察,因而本文从另一个角度处理该问题,我们的算法的基本思想如下:首先假设所有的状态变量都不是观察变量,这时部分可观察的强规划问题就被转换为一个一致性规划问题.求解该一致性规划问题可能会得到两种结果:如果规划求解成功,那么意味着该问题不需要任何观察变量就可以求得规划解;如果求解不

成功,那么意味着该问题需要增加必要的观察变量来区分一致性规划求解过程中所得到的失败的信念状态.在第 2 种情况下,我们逐步增加必要的观察变量,并将区分后的信念状态作为初始状态重新求解,重复上述过程直到初始条件树的根节点被标记为 *Success*,或者所有的状态变量都被加入到约减的观察变量集合,或者无用的观察变量集合的个数等于所有状态变量的个数减 1 为止.这时我们就可以得到约减的观察变量.图 2 给出了算法的主体框架.

```
1  function DECREASEOBS(I,G,X)
2      O_deo:=∅; result:=false;
3      ST:=MKINITIALTREE(I);
4      IT:=MKINITIALTREE(I);
5      if (I⊆G) then
6          return O_deo;
7      endif
8      TAGSTNODE(ROOT(ST),Undetermined);
9      while (¬ISSUCC(ROOT(IT))∨O_deo≠O_all∨|O_no|≠|O_all|−1) do
10         result=CONFORMANT(BEL(andnode(IT)),G,ST,O_deo,X,IT);
11         if (result==true)
12         then PROPAGATESUCCIT(leaves(IT),IT);
13         else IDENTIFYBEL(ST,O_deo,X,IT)
14         endif
15     endwhile
16     return O_deo;
17 end
```

Fig.2 Decrease observation variables algorithm

图 2 约减观察变量算法

约减观察变量算法假设所有的状态变量都不是观察变量,即 $O_{deo}=\varnothing$,在算法中我们将使用两个重要的数据结构,即搜索树和初始条件树.搜索树的根节点为初始状态节点.在扩展搜索树时,新生成的节点根据其不同的表示意义而带有不同的标记,分别为 *Success*,*Failure*,*Examined* 和 *Undetermined*.其中 *Success* 表示已经找到一条到达目标状态的一致性规划;*Failure* 表示该节点的信念状态没有可用动作,这时会导致当前一致性规划无解;*Examined* 表示节点已经被考察过;其他尚未标记的节点均为 *Undetermined* 节点.我们将所有的 *Undetermined* 节点构成的集合称为边界节点集合,待扩展的节点就是从边界节点集合中选出的.初始条件树可以被看作是一棵与或树,根节点为初始状态节点,其他节点由各个搜索树中的 *Failure* 节点和根节点组成.其中与节点由同一个 *Failure* 节点所区分的两个信念状态所形成的节点构成,或节点由一致性规划求解过程中所形成的 *Failure* 节点构成.

首先,我们初始化搜索树和初始条件树(MKINITIALTREE),并判断该部分可观察强规划问题的初始状态是否包含在目标状态中.如果是,则算法结束,返回一个空的约减观察变量集合;否则,算法将搜索树中的初始状态节点标记为 *Undetermined*(*TAGSTNODE*),同时把该部分可观察的强规划问题转化为一致性规划问题,进入第 9~15 行的循环过程.在循环过程中,算法会调用 CONFORMANT 子过程生成一致性规划解,且一致性规划问题的初始状态是初始条件树中与节点所对应的信念状态.如果规划求解成功,那么算法调用 PROPAGATESUCCIT 子过程在初始条件树中自底向上传播成功信息;否则,算法调用 IDENTIFYBEL 子过程增加必要的观察变量来区分一致性规划求解过程中所得到的失败的信念状态.重复上述过程,直到初始条件树的根节点被标记为 *Success* 或所有的状态变量都被添加到约减的观察变量集合或无用的观察变量集合的个数等于所有状态变量的个数减 1 为止,在算法结束时,O_{deo} 集合中的观察变量即为约减的观察变量.

如上所述,在我们的算法中存在 3 个重要的子过程:CONFORMANT 子过程,IDENTIFYBEL 过程和 PROPAGATESUCCIT 子过程,这 3 个子过程之间的联系如图 3 所示.CONFORMANT 子过程用于对一致性规划

问题进行求解,这部分的内容主要在本文的第 3 节介绍.IDENTIFYBEL 子过程用于区分失败的信念状态并逐步增加必要的观察变量,我们将在第 4 节进行介绍.PROPAGATESUCCIT 子过程用来自底向上地在初始条件树中传播成功信息,我们将在第 5 节中给出介绍.

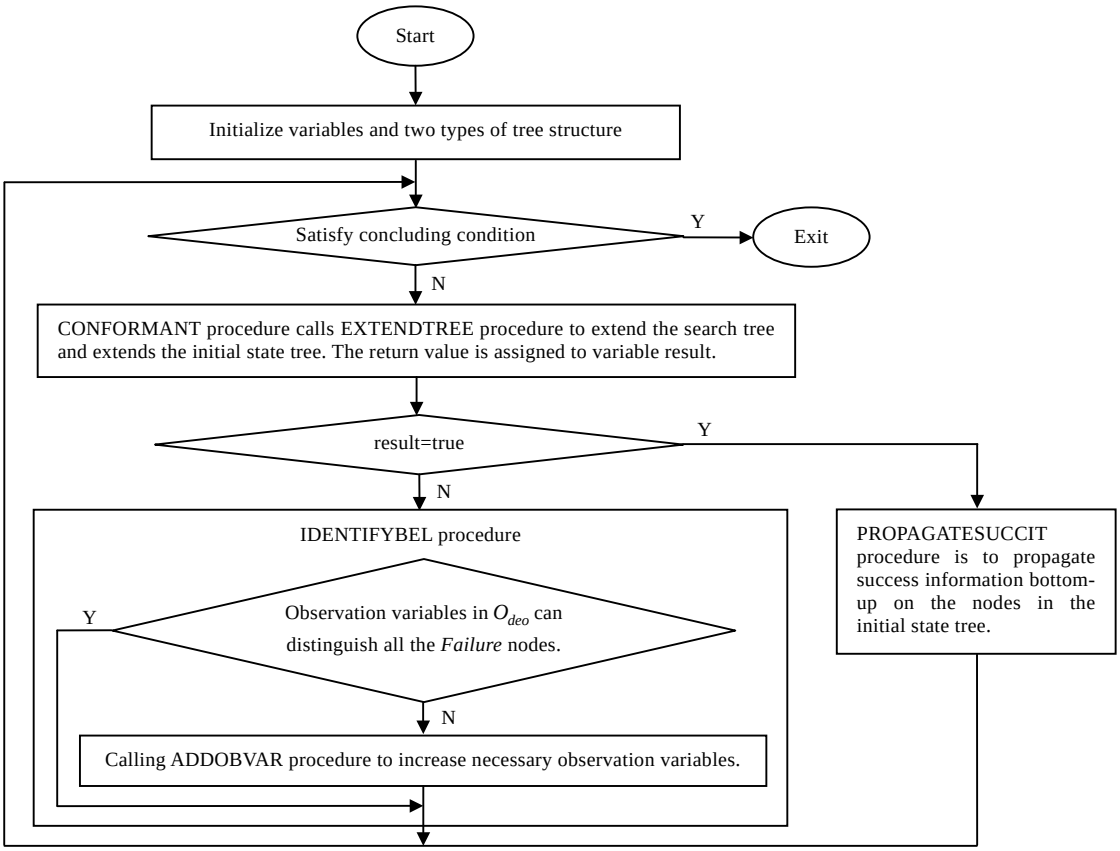


Fig.3 Sketch map of decrease observation variables algorithm

图 3 约减观察变量算法示意图

3 生成一致性规划

生成一致性规划子过程在一致性规划解存在的情况下用来找到规划解,在一致性规划解不存在的情况下用来找到失败的信念状态.图 4 给出了求解一致性规划子过程的基本步骤.

CONFORMANT 子过程首先根据节点的可用动作,调用 EXTENDTREE 子过程扩展搜索树并对新生成的节点进行标记,直到边界节点集合为空或者搜索树中存在一个被标记为 *Success* 的节点为止(第 2~5 行),此时表明这棵搜索树已经扩展结束.然后该过程会判断搜索树中是否存在标记为 *Success* 的节点,如果存在这样的节点,就意味着该规划问题从初始状态可以找到一致性规划解,此时我们把初始条件树中的该一致性规划问题的初始状态节点标记为 *Success*(第 6~8 行);否则,我们就会找到导致不能生成一致性规划解的失败的信念状态,即搜索树中被标记为 *Failure* 的节点,并将这些 *Failure* 节点和或边加入到初始条件树中(第 9~12 行).

EXTENDTREE 子过程用来扩展搜索树并对搜索树中各个节点进行标记.图 5 给出了扩展搜索树子过程的基本算法.

```
1   procedure CONFORMANT(I,G,ST,O_deo,X,IT)
2     while (FRONTIER(ST)≠∅∧SUCCESSNODE(ST)=∅) do
3           node:=EXTRACTNODEFROMFRONTIER(ST);
4           EXTENDTREE(node,ST);
5     endwhile
6     if (SUCCESSNODE(ST)≠∅) then
7           TAGITNODE(NODE(I),Success);
8           return true;
9     else
10          Nodes(IT):=Nodes(IT)∪FAILURENODES;
11          Arcs(IT):=Arcs(IT)∪⟨initialST(IT),FAILURENODE⟩;
12          return false;
13    endif
14  end
```

Fig.4 CONFORMANT procedure

图 4 生成一致性规划子过程

```
1   procedure EXTENDTREE(n,ST)
2     if nonapplication(BEL(n), all a∈A) then TAGNODE(n,Failure);
3     else
4       forall a∈A
5         if applicable(BEL(n),a) then
6           n′:=NODE(R(BEL(n),a));
7           Nodes(ST):=Nodes(ST)∪{n′};
8           Arcs(ST):=Arcs(ST)∪⟨n,a,n′⟩;
9           if (BEL(n′)⊆G) then TAGSTNODE(n′,Success);
10          else if (∃n_a∈ST:BEL(n′)=BEL(n_a)) then TAGSTNODE(n′,Examined);
11          else TAGSTNODE(n′,Undetermined);
12          endif
13          TAGSTNODE(n,Examined);
14        endif
15      endfor
16    endif
17    forall n′∈NODEUNDETERMINED(ST)
18      forall a∈A
19          if applicable(BEL(n′),a) then n″:=NODE(R(BEL(n′),a));
20          endif
21      endfor
22      if (BEL(SONS(n′))==BEL(FATHERS(n′))) then TAGSTNODE(n′,Failure);
23      endif
24    endfor
25  end
```

Fig.5 EXTENDTREE procedure

图 5 扩展搜索树子过程

EXTENDTREE 子过程首先判断待扩展的节点是否有可用动作,如果没有,那么标记该节点为 *Failure*(第 2 行);否则,根据该节点的可用动作把生成的节点和边加入到搜索树中(第 5~8 行),然后对生成的节点进行标记(第 9~25 行).具体做法为:如果生成的节点的信念状态包含于目标状态中,那么标记该节点为 *Success*(第 9 行);如果生成的节点与已经生成的节点相同,那么标记该节点为 *Examined*(第 10 行),并把所有未被标记的节点标记为

Undetermined(第 11 行).最后,对标记为 *Undetermined* 的节点进行考察(第 17~24 行),如果标记为 *Undetermined* 节点的信念状态通过所有可用动作所到达的信念状态与该节点所有的父节点的信念状态完全相同,就把该节点标记为 *Failure*.

生成一致性规划子过程实际上就是搜索树的扩展并标记的过程,下面我们用例 2 说明如何建立搜索树.

例 2:考虑例 1 介绍的机器人导航问题,设它的初始状态为 s_0 或 s_3,即 $I=\{s_0,s_3\}$,目标状态为 s_6,即 $G=\{s_6\}$(为了作图方便,我们对搜索树中相同的节点只生成 1 次,所以搜索树中会出现环).

初始时,初始条件树和搜索树都只由初始状态节点组成.在生成一致性规划子过程中,首先判断循环条件,此时边界节点集合包含搜索树的根节点,且标记为 *Success* 的节点集合为空,搜索树满足循环条件,进入第 1 次循环过程.按照一定的策略从边界节点集合中挑选$\{s_0,s_3\}$作为待扩展的节点,调用 EXTENDTREE 子过程.因为信念状态$\{s_0,s_3\}$只有 1 个可用动作 *GoEast*,根据状态转移函数 $R(\{s_0,s_3\},GoEast)=\{s_1,s_4,s_7\}$ 把 $\{s_1,s_4,s_7\}$ 节点和边加入到搜索树中.因为新加入的节点所对应的信念状态不包含在目标状态中,并且搜索树中不存在与该节点一致的节点,所以该节点被标记为 *Undetermined*.接下来考察所有被标记为 *Undetermined* 的节点,现在被标记为 *Undetermined* 的节点只有 $\{s_1,s_4,s_7\}$,而信念状态 $\{s_1,s_4,s_7\}$ 的可用动作是 *GoEast* 和 *GoSouth*,根据状态转移函数可知,该信念状态可以生成的儿子节点所对应的信念状态并不与它所有父节点所对应的信念状态相同,因而 $\{s_1,s_4,s_7\}$ 节点不会被标记为 *Failure*.注意,此时我们并没有把生成的$\{s_2,s_5,s_8\}$,$\{s_4,s_7,s_{10}\}$节点加入到搜索树中,这一步只是为了判断节点 n' 应带有哪种类型的标记.现在第 1 次循环过程执行结束.此时得到的搜索树如图 6 所示.重复上述方式,算法进行了 4 次循环过程,最后所建立的搜索树如图 7 所示.

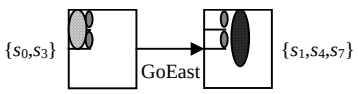


Fig.6 Search tree at the first cycle

图 6 第 1 次循环所生成的搜索树

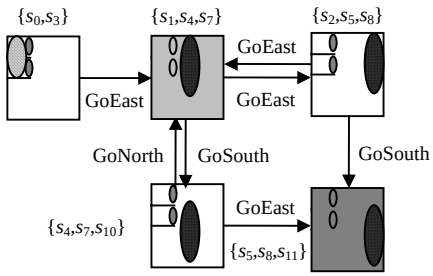


Fig.7 The first search tree

图 7 第 1 棵搜索树

4 信念状态区分

当一致性规划求解失败时,需要对失败的信念状态加以区分.图 8 给出了区分信念状态子过程的基本算法.

在该过程中,*OBSSTFAILURE* 表示原有 O_{deo} 集合中的观察变量不能区分的 *Failure* 节点的集合;*BEL*(*n*)表示存储 *Failure* 节点 *n* 的数组,假设它有 *k* 个元素,数组中的各个元素分别对应于该节点信念状态中的各个状态;*MARKNODE* 记录用现有 O_{deo} 中的观察变量可以区分的 *Failure* 节点的数量,这里假设有 *t* 个 *Failure* 节点;*MARKDIS* 记录约减的观察变量区分失败信念状态的次数.

该子过程首先用 O_{deo} 集合中的观察变量区分所有 *Failure* 节点的信念状态(第 3~23 行).具体做法为:对每一个 *Failure* 节点,看它的信念状态是否可以区分为一个状态和一个不包含该状态的信念状态.如果可以,即 *MARKDIS* 的值为 *k*–1,则增加 *MARKNODE* 的值,并把区分的两个信念状态对应的节点和与边加入到初始条件树中;否则,把该 *Failure* 节点加入到 *OBSSTFAILURE* 集合中.然后,判断 *MARKNODE* 的值是否为 *t*.如果是,则把 *OBSSTFAILURE* 集合置空.同样的过程,继续考察其他 *Failure* 节点.然后检查 *OBSSTFAILURE* 集合(第 24~26 行).如果该集合为空,则表明 O_{deo} 集合中的观察变量可以区分所有失败的信念状态;否则,表明 O_{deo} 集合中的观察变量不能区分该集合中所有的 *Failure* 节点,此时,需要调用 *ADDOBVAR* 子过程来逐步增加必要的观察变量.

```
1   procedure IDENTIFYBEL(ST,O_deo,X,IT)
2       OBSSTFAILURE:=∅; MARKNODE:=0;
3       forall n∈NODEFAILURE(ST)
4           for m:=1; k≥m≥1; m++
5               MARKDIS:=0;
6               for p:=1; k≥p≥1; p++
7                   if (X(BEL(n)[m],O_deo)≠X(BEL(n)[p],O_deo)) then
8                       MARKDIS:=MARKDIS+1;
9                   endif
10              endfor
11              if (MARKDIS==k–1) then
12                  MARKNODE:=MARKNODE+1;
13                  Nodes(IT)=Nodes(IT)∪NODE(BEL(n)[m]);
14                  Nodes(IT)=Nodes(IT)∪NODE(BEL(n)–BEL(n)[m]);
15                  Arcs(IT)=Arcs(IT)∪⟨n,BEL(n)[m],BEL(n)–BEL(n)[m]⟩;
16                  goto 4
17              else continue
18              endif
19          endfor
20          OBSSTFAILURE:=OBSSTFAILURE∪{n};
21          if (MARKNODE==t) then OBSSTFAILURE:=∅;
22          endif
23      endfor
24      if (OBSSTFAILURE≠∅) then ADDOBVAR(OBSSTFAILURE,O_deo,X,IT);
25      endif
26  end
```

Fig.8 IDENTIFYBEL procedure

图 8 区分信念状态子过程

4.1 增加观察变量子过程

增加观察变量子过程用于逐步增加必要的观察变量.该子过程分两种情况给出:当存在一个观察变量可以区分规划域中任意两个状态时,算法得到一个最小的观察变量集合;当不存在这样一个观察变量时,算法得到一个尽可能小的观察变量集合,但是,我们不能保证该集合最小.

4.1.1 增加观察变量子过程(第 1 种情况)

因为存在一个观察变量可以区分规划域中任意两个状态,所以该子过程只能对每一个观察变量进行考察,以找到一个必要的观察变量.图 9 给出了区分信念状态子过程(第 1 种情况)的基本算法.

在该过程中,*BEL*(*n*)表示存储 *Failure* 节点 *n* 的数组,假设它有 *k* 个元素,数组中的各个元素对应于该节点信念状态中的各个状态;*MARKNODE* 用来记录 O_{deo} 集合中观察变量可以区分 *Failure* 节点的数量;*MARKDIS* 用来记录 O_{deo} 集合中的观察变量区分失败信念状态的次数;O_{no} 集合用来记录所有考察过的、无用的观察变量.

该过程首先初始化 O_{deo} 和 O_{no} 集合(第 2 行),然后处理初始条件树,删除初始条件树中除根节点和第 1 棵搜索树中的 *Failure* 节点以外的所有节点,以便用新的观察变量重新对它们加以区分(第 3~4 行).最后考察每一个不属于 O_{no} 集合中的状态变量,并把该状态变量看作观察变量(第 5~29 行).具体做法为:首先判断该观察变量是否可以区分初始条件树中根节点的每一个儿子节点(假设根节点有 *t* 个儿子)对应的信念状态,即第 1 棵搜索树中 *Failure* 节点对应的信念状态(第 7~28 行),对每一个儿子判断该观察变量是否可以把儿子节点的信念状态区分为一个状态和一个不包含该状态的信念状态.如果可以,即 *MARKDIS* 的值为 *k*–1,则增加 *MARKNODE* 的值并把生成的节点和与边加入到初始条件树中(第 8~23 行).同样的过程,继续考察其他儿子节点.当所有节点都被考察完毕后,会出现两种结果:如果存在不可以用该观察变量区分的儿子节点,即 *MARKNODE* 的值不为 *t*,则把该观察变量加入到 O_{no} 集合(第 24 行);否则,加入到 O_{deo} 集合(第 25 行).同样的过程,继续考察其他状态变量,直到找到一个必要的观察变量为止.

```
1  procedure ADDOBVAR(OBSSTFAILURE,O_deo,X,IT)
2      O_no:=O_deo;O_deo:=∅;
3      if (grandsons(ROOT(IT))≠∅) then prune grandsons(ROOT(IT))
4      endif
5      forall o_m∉O_no
6          MARKNODE:=0;
7          forall n∈SONS(ROOT(IT))
8              for m:=1; k≥m≥1; m++
9                  MARKDIS:=0;
10                 for p:=1; k≥p≥1; p++
11                     if (X(BEL(n)[m],o_m)≠X(BEL(n)[p],o_m)) then
12                         MARKDIS:=MARKDIS+1;
13                     endif
14                 endfor
15                 if (MARKDIS==k−1) then
16                   MARKNODE:=MARKNODE+1;
17                   Nodes(IT)=Nodes(IT)∪NODE(BEL(n)[m]);
18                   Nodes(IT)=Nodes(IT)∪NODE(BEL(n)−BEL(n)[m]);
19                   Arcs(IT)=Arcs(IT)∪⟨n,BEL(n)[m],BEL(n)−BEL(n)[m]⟩;
20                   goto 9
21                 else continue
22                 endif
23             endfor
24             if (MARKNODE≠t) then O_no:=O_no∪{o_m}; goto 4
25             else O_deo:={o_m}; goto 36
26             endif
27         endfor
28     endfor
29 end
```

Fig.9 ADDOBVAR procedure (the first case)

图 9 增加观察变量子过程(第 1 种情况)

定理 1. 设存在一个观察变量 o 可以区分规划域中任意两个状态,则约减观察变量算法(第 1 种情况)得到一个最少的观察变量集合 O_{deo}.

证明:(1) 若部分可观察强规划问题不需要任何观察变量就可以求得规划解,则根据约减观察变量算法,存在两种情况:① 若 $I \subseteq G$,则该规划不需要任何动作就可以到达目标状态,算法直接返回 $O_{deo}=\varnothing$.② 若 $I \not\subset G$ 时,则约减观察变量算法会调用一次生成一致性规划子过程,并且该子过程一定能到达目标状态.因为整个算法并没有向 O_{deo} 集合中添加任何变量,所以返回 $O_{deo}=\varnothing$.因此,无论对情况①还是情况②来说,它们得到的观察变量的数量都是最少的.(2) 若部分可观察强规划问题需要观察变量才能使问题得到求解,显然,只要证明与观察变量有关的区分信念状态子过程在区分失败的信念状态时,O_{deo} 集合一直只存在 1 个观察变量就足够了,即 $|O_{deo}|=1$.假设通过生成一致性规划子过程得到了若干个标记为 *Failure* 的节点,现在用 O_{deo} 中的观察变量对这些 *Failure* 节点所对应的信念状态进行区分:① 显然,对于不用调用增加观察变量子过程的情况,O_{deo} 集合不会改变,只要说明上次调用增加观察变量子过程中 O_{deo} 集合里只有 1 个元素就可以了.② 现在着重讨论需要调用增加观察变量子过程的情况.首先,该子过程初始化 O_{deo} 集合,使 $O_{deo}=\varnothing$.然后,选择一个状态变量作为观察变量对所有 *Failure* 节点进行区分.如果该观察变量不可以区分所有失败的信念状态,则把它放在集合 O_{no} 中,O_{deo} 集合不变;否则,把该观察变量加入到 O_{deo} 集合中,此时 $|O_{deo}|=1$,重复上述过程直到找到这样一个观察变量为止.现在可以看出,只要调用增加观察变量子过程,得到的约减的观察变量集合始终只有 1 个元素.因此,无论对情况①还是情况②,约减的观察变量集合中一直只存在 1 个观察变量.故它得到的观察变量的集合是最少的.综上所述,约减观察变量算法(第 1 种情况)可以得到一个最少的观察变量集合.

4.1.2 增加观察变量子过程(第 2 种情况)

由于不存在一个观察变量可以区分规划域中任意两个状态,因此只能在原有约减观察变量集合的基础上逐步增加必要的观察变量.图 10 给出了区分信念状态子过程(第 2 种情况)的基本算法.

```
1  procedure ADDOBVAR(OBSSTFAILURE,O_deo,X,IT)
2      forall o_m∉O_deo
3          O_deo:=O_deo∪{o_m}; MARKNODE:=0;
4          forall n∈OBSSTFAILURE
5              for m:=1; k≥m≥1; m++
6                  MARKDIS:=0;
7                  for p:=1; k≥p≥1; p++
8                      if (X(BEL(n)[m],O_deo)≠X(BEL(n)[m],O_deo) (BEL(n)[p],O_deo)) then
9                          MARKDIS:=MARKDIS+1;
10                     endif
11                 endfor
12                 if (MARKDIS==k−1) then
13                     MARKNODE:=MARKNODE+1;
14                     Nodes(IT)=Nodes(IT)∪NODE(BEL(n)[m]);
15                     Nodes(IT)=Nodes(IT)∪NODE(BEL(n)−BEL(n)[m]);
16                     Arcs(IT)=Arcs(IT)∪⟨n,BEL(n)[m],BEL(n)−BEL(n)[m]⟩;
17                     goto 5
18                 else continue
19                 endif
20             endfor
21             if (MARKNODE≠r) then O_deo:=O_deo−{o_m}; goto 2
22             else goto 31
23             endif
24         endfor
25     endfor
26 end
```

Fig.10 ADDOBVAR procedure (the second case)

图 10 增加观察变量子过程(第 2 种情况)

该子过程中变量的含义与第 1 种情况相同.该过程考察每一个不在 O_{deo} 集合中的状态变量(第 2~25 行).首先把它看作观察变量加入到 O_{deo} 集合中(第 3 行),然后分析 O_{deo} 集合中所有的观察变量是否可以区分 *OBSSTFAILURE* 集合中的每一个 *Failure* 节点的信念状态(假设 *OBSSTFAILURE* 集合中有 r 个元素)(第 4~24 行).如果可以,即 *MARKDIS* 的值为 k–1,那么就增加 *MARKNODE* 的值并把生成的节点和与边加入到初始条件树中(第 12~16 行).同样的过程,继续考察 *OBSSTFAILURE* 集合中的其他节点.当所有的节点都被考察完毕后,会出现两种结果:如果该集合中存在用添加后的观察变量集合不可区分的节点,即 *MARKNODE* 的值不为 r,则把该状态变量从约减的观察变量集合中删除(第 21 行);否则,保留该状态变量.同样的过程,继续考察其他状态变量,直到找到能够区分所有 *Failure* 节点的观察变量集合为止.

例 3:考虑例 1 介绍的机器人导航问题.该问题中可能的观察变量有 *WallE*,*WallW*,*WallS*,*WallN*.由图 1 可以看出,状态 s_5 和 s_8 的观察函数相同,即 $X(s_5,o_i)=X(s_8,o_i)$,不存在一个观察变量可以区分域中任意两个状态,所以文献[7]不能求解,而本文给出的算法(第 2 种情况)却可以处理该情况.下面我们以例 3 来说明区分信念状态子过程.

对例 2 生成的 *Failure* 节点$\{s_5,s_8,s_{11}\}$用 O_{deo} 集合中的观察变量进行区分.因为到目前为止没有任何观察变量加入到 O_{deo} 集合,故不能区分所有失败的信念状态,需调用增加观察变量子过程来增加必要的观察变量.在增加观察变量子过程中,首先根据变量的意义确定变量的值:r=1,k=3.然后任意选择一个状态变量 *WallS* 作为观察变量并加入到 O_{deo} 集合中.接下来用 O_{deo} 集合中的观察变量对每一个 *Failure* 节点进行考察.首先进入外层 for 循环,此时 m=1,对变量 *MARKDIS* 初始化,进而进入循环体中的 for 循环,因为 $X(s_5,WallS)\neq X(s_{11},WallS)$,所以,当

p=3 时,里层循环满足判断条件,变量 *MARKDIS* 执行加 1 操作,里层循环执行结束.接下来不满足 *MARKDIS*=k−1 的判断条件,继续执行外层的 for 循环.我们发现当 m=2 时所得到的结果与 m=1 时的完全相同.但是,当 m=3 时,因为 $X(s_{11},WallS)\neq X(s_5,WallS)$,$X(s_{11},WallS)\neq X(s_8,WallS)$满足判断条件,变量 *MARKDIS* 连续两次加 1,所以里层循环结束时变量 *MARKDIS* 的值为 2,正好满足 *MARKDIS*=(k−1=2)的判断条件,这时把信念状态$\{s_{11}\}$和信念状态$\{s_5,s_8\}$所形成的节点和与边加入到初始条件树中,并对变量 *MARKNODE* 执行加 1 操作,现在外层循环执行结束.此时判断 *MARKNODE* 的值是 1,不满足判断条件 *MARKNODE*≠(r=1),子过程执行结束.此时的初始条件树如图 11 所示.

下面用例 4 说明“当不存在一个观察变量可以区分规划域中任意两个状态时,我们的约减观察变量算法(第 2 种情况)不能保证所求的观察变量集合最少”的问题.

例 4:图 12 给出了一个类似于例 1 的简单机器人导航问题,它包含 4×3 个房间,分别为 $s_0,s_1,...,s_{11}$.机器人可位于域中的任意一个房间,即有限的状态集合为 $S=\{s_0,s_1,...,s_{11}\}$;机器人可以向 4 个方向移动,即有限的动作集合为 A={*GoNorth*,*GoSouth*,*GoEast*,*GoWest*}.另外,动作的可用性与例 1 的相同.

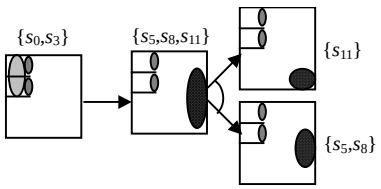


Fig.11 Initial tree after executing the ADDOBVAR procedure

图 11 调用一次增加观察变量子过程后的初始条件树

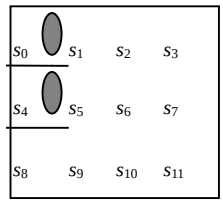


Fig.12 Another simple robot navigation problem

图 12 另一个简单的机器人导航问题

对于该问题来说,其可能的观察变量有 *WallE*,*WallW*,*WallS*,*WallN*.根据图 12 可以看出信念状态$\{s_5,s_9\}$,$\{s_6,s_{10}\}$的观察函数值完全相同,所以该例对应于本文算法的第 2 种情况.假设通过生成一致性规划子过程,得到的失败的信念状态是$\{s_3,s_7,s_{11}\}$,而此时 O_{deo}=∅,我们会调用增加观察变量子过程,在该子过程中加入了 *WallN*,可以看出 *WallN* 能够把信念状态区分开.接下来继续以这两个区分的信念状态作为一致性规划的初始状态扩展搜索树,假设我们又得到了失败的信念状态$\{s_5,s_9\}$,此时 O_{deo} 集合中的 *WallN* 不能区分该信念状态,需要调用增加观察变量子过程逐步增加观察变量,在该子过程中加入了 *WallS*,发现此时的 O_{deo} 可以区分失败的信念状态.我们又以区分出的信念状态作为一致性规划的初始状态生成一致性规划,假设它们可以找到规划解,此时 O_{deo}={*WallN*,*WallS*}就是约减的观察变量集合.实际上,我们发现当 O_{deo}={*WallS*}时就可以完全区分这两个失败的信念状态$\{s_3,s_7,s_{11}\}$和$\{s_5,s_9\}$,而通过我们的算法,O_{deo} 集合中却有两个元素,因而我们的算法(第 2 种情况)不能保证所求的观察变量集合最少.

5 成功信息传播

传播成功信息子过程用来自底向上地在初始条件树中的各个节点上传播成功信息.图 13 具体描述了该子过程.

该子过程首先对变量 *res* 进行赋值(第 2~5 行).具体做法为:如果被考察节点有一对与节点儿子被标记为 *Success* 或有一个或节点儿子被标记为 *Success*,变量 *res* 就被赋为 true(第 3~5 行).如果变量 *res* 的值为 true,则把当前节点标记为 *Success*.然后,把当前节点的父节点作为待考察的节点(第 8~9 行),重复该过程直到被考察的节点是初始条件树的根节点为止(第 6~12 行).

例 5:我们将通过例 1 介绍的机器人导航问题来整体说明约减观察变量算法.

首先假设所有的状态变量都不是观察变量,即 O_{deo}=∅.根据例 1 可知,$I=\{s_0,s_3\}\subseteq G=\{s_6\}$.因此,我们把该部分可观察的强规划问题转化为一个一致性规划问题进行搜索树的建立.此过程的具体细节见例 2,所得结果如图 7

所示(浅灰色表示 *Examined* 节点,灰色表示 *Failure* 节点,深灰色表示 *Success* 节点.为了作图方便,我们对搜索树中部分相同的节点只生成 1 次,因而在搜索树中会出现环).

```
1  procedure PROPAGATESUCCIT(n,IT)
2    res:=false;
3    if (onepairandsons(n)=Success)∨(oneorson(n)=Success) then
4      res:=true;
5    endif
6    if (n≠ROOT(IT)) then
7      if (res==true) then
8          TAGITNODE(n,Success);
9          n_father:=FATHER(n);
10          PROPAGATESUCCIT (n_father,IT);
11     endif
12   endif
13 end
```

Fig.13 PROPAGATESUCCIT procedure

图 13 传播成功信息子过程

从图 7 可以看到该搜索树中存在一个 *Failure* 节点{s_5,s_8,s_{11}},这就意味着需要调用区分信念状态子过程区分失败的信念状态.该过程已经在例 3 中给出了详细的介绍,它的执行结果是该 *Failure* 节点被 O_{deo}={*WallS*}区分为信念状态{s_{11}}和{s_5,s_8},此时生成的初始条件树如图 11 所示.因为此时初始条件树的根节点没有被标记为 *Success*,故算法第 2 次调用生成一致性规划子过程,它们分别以{s_{11}}和{s_5,s_8}作为一致性规划问题的初始状态寻找一致性规划解,所建立的搜索树如图 14 所示(因为空间有限,这里只对部分节点进行了扩展),此时的初始条件树如图 15 所示.

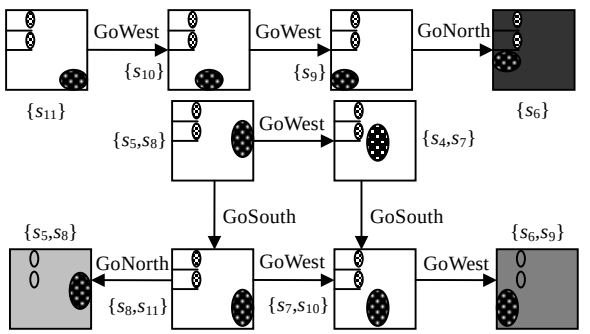


Fig.14 The second and third search tree

图 14 第 2 棵和第 3 棵搜索树

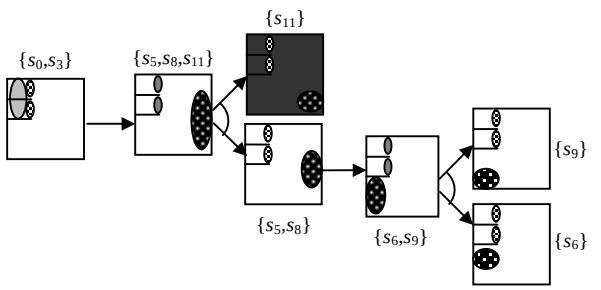


Fig.15 Initial tree after establishing the second and third search tree

图 15 第 2 棵和第 3 棵搜索树建立后的初始条件树

由图 14 可以看出,以$\{s_{11}\}$为初始状态的一致性规划问题可以找到一致性规划解,此时,把$\{s_{11}\}$在初始条件树中所对应的节点标记为 *Success*.以$\{s_5,s_8\}$为初始状态的一致性规划问题在寻找一致性规划解的过程中出现了 *Failure* 节点,这就意味着需要对该节点的信念状态进行区分.此过程所得结果如下:用O_{deo}={*WallS*}可以把该信念状态区分为$\{s_6\}$和$\{s_9\}$,因而不需要调用 ADDOBVAR 子过程.现在回到主函数中,接下来执行传播成功信息子过程.从图 15 可以看到此时的初始条件树中存在一个被标记为 *Success* 的节点$\{s_{11}\}$,但是$\{s_{11}\}$节点是与节点,只有当它的兄弟节点也被标记为 *Success* 时,变量 *res* 才为 true,进而它的根节点才可能被标记为 *Success*.因而算法第 3 次调用生成一致性规划子过程,所得结果如图 16 所示.

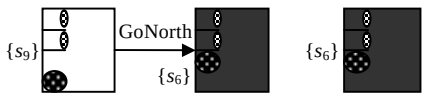


Fig.16 The fourth and fifth search tree

图 16 第 4 棵和第 5 棵搜索树

从图 16 容易看出,以$\{s_6\}$和$\{s_9\}$作为初始状态的一致性规划问题都可以找到规划解.因为这两棵搜索树中都存在 *Success* 节点,因而,这两个规划问题在初始条件树中对应的初始状态节点被标记为 *Success*.回到主函数中调用传播成功信息子过程.因为$\{s_6,s_9\}$节点的一对与儿子节点都被标记为 *Success*,所以变量 *res* 被赋值为 true,进而把$\{s_6,s_9\}$节点标记为 *Success*,并把当前节点的父节点作为下次要考察的节点,递归地调用该过程,直到初始条件树的根节点被标记为 *Success* 时结束.经过此过程,最后,初始条件树的根节点$\{s_0,s_3\}$被标记为 *Success*,算法结束,O_{deo}={*WallS*}即为所求的约减的观察变量集合.

6 约减观察变量算法与文献[7]中算法的比较

本文和文献[7]都给出了求解约减观察变量的方法,下面我们从以下几个方面比较这两种算法.

(1) 从算法策略上考虑.文献[7]中的算法采用的是一种删除无用观察变量的策略,对于给定的一个强规划问题,该算法首先假设所有状态变量为观察变量,在此基础上逐步删除不必要的观察变量,最终得到最小的必要观察变量集合.而本文的约减观察变量算法采用的是一种增加有用观察变量的策略,我们首先假设所有的状态变量都不是观察变量,在此基础上逐步增加必要的观察变量,从而最终得到一个最小的必要观察变量集合.因此,给定一个强规划问题,如果它的必要观察变量较多,则文献[7]中的方法效率更高;反之,如果它的必要观察变量较少,则我们的方法更为适合.从这个意义上说,我们的方法与文献[7]中的方法是互补的.

(2) 从算法的局限性上考虑.文献[7]中的算法在执行过程中存在一个约束:假设存在一个观察变量能够区分规划问题中的任意两个状态.而本文的约减观察变量算法则不要求此约束条件,当规划问题中存在一个这样的观察变量时,本文的算法可以给出最少的观察变量集合;当规划问题中不存在一个这样的观察变量时,本文的算法也可以得到约减的观察变量,只是不能保证观察变量集合最小.另外,文献[7]中的算法要求所有状态变量为观察变量,然而在很多情况下,我们无法对所有的状态变量进行观察,因此我们对此不作要求.

(3) 从算法的结果上考虑.当不存在一个观察变量能够区分规划问题中任意两个状态时,文献[7]中的算法不可求解,而本文的约减观察变量算法虽然不能保证约减的观察变量集合最小,但依然可以求得该集合.

7 结 论

目前,在部分可观察规划的研究中,观察变量都是静止不变的.事实上,设定观察变量的意义在于获得信息,而任何信息的获取都需要一定的代价.因而给定一个部分可观察强规划问题,由计算机自动确定一个最小(或尽可能小)的必要观察变量集合是非常有必要的.为此,本文给出了一种约减观察变量方法——假设所有的状态变量都不是观察变量,在此基础上逐步增加必要的观察变量,从而最终得到一个必要的观察变量集合.这种方法可以弥补文献[7]中算法的限制——必须找到部分可观察规划问题在完全可观察环境下强规划,使其具有更好的通用性.考虑到文献[7]中的假设:至少有 1 个观察变量能够区分域中任意两个状态的局限性,我们的约减观察变

量算法分两种情况给出:当存在一个观察变量可以区分规划域中任意两个状态时,算法得到一个最小的观察变量集合;当不存在这样一个观察变量时,算法得到一个尽可能小的观察变量集合,但是我们不能保证该集合最小.对于算法的时间复杂性问题,由于部分可观察的强规划问题本身就是一个指数级别的问题[18,19],所以无论是本文给出的算法还是文献[7]给出的算法,其时间复杂性在最坏情况下都是指数级别的.目前,随着一致性规划和强规划求解效率的提高,文献[7]中的算法在 1 秒内已经可以处理包含 10^{10} 以上状态的规划问题,而我们的算法的求解效率直接依赖于一致性规划和强规划的求解,因此,对于算法可处理的规模问题,我们的算法具有类似的求解效率.

今后我们将从以下几个方面开展研究工作:

(1) 在本文中,感知器所感知的信息都是真实的,即不存在误导观察,今后的工作将扩展到此方面.

(2) 考虑到不同观察变量获得信息所需的代价不同,以后我们将从代价的方面考虑如何减少观察变量.

(3) 在不确定规划域中有这样一组规划:弱规划、强规划、强循环规划[20,21].在本文中,我们研究了部分可观察的强规划,以后我们将把部分可观察的工作扩展到弱规划和强循环规划中.

References:

[1] Weld DS. Recent advances in AI planning. AI Magazine, 1999,20(2):93−123.

[2] Bertoli P, Cimatti A, Roveri M, Traverso P. Planning in nondeterministic domains under partial observability via symbolic model checking. In: Nebel B, ed. Proc. of the IJCAI. Seattle: Morgan Kaufmann Publishers, 2001. 473−478.

[3] Bertoli P, Cimatti A, Roveri M, Traverso P. Strong planning under partial observability. Artificial Intelligence, 2006,170(4-5): 337−384.

[4] Bertoli P, Cimatti A, Traverso P. Interleaving execution and planning for nondeterministic, partially observable domains. In: de Mántaras RL, Saitta L, eds. Proc. of the ECAI. Amsterdam: IOS Press, 2004. 657−661.

[5] Herzig A, Lang J, Marquis P. Action representation and partially observable planning using epistemic logic. In: Gottlob G, Walsh T, eds. Proc. of the Int'l Joint Conf. on Artificial Intelligence (IJCAI). Morgan Kaufmann Publishers, 2003. 1067−1072.

[6] Poupart P, Boutilier C. Vector-Space analysis of belief-state approximation for POMDPs. In: Nebel B, ed. Proc. of the Conf. on Uncertainty in Artificial Intelligence (IJCAI). Seattle: Morgan Kaufmann Publishers, 2001. 445−452.

[7] Huang W, Wen ZH, Jiang YF, Wu LH. Observation reduction for strong plans. In: Veloso MM, ed. Proc. of the Int'l Joint Conf. on Artificial Intelligence (IJCAI). Menlo Park: AAAI Press, 2007. 1930−1935.

[8] Bertoli P, Cimatti A, Roveri M. Heuristic search+symbolic model checking=efficient conformant planning. In: Nebel B, ed. Proc. of the 7th IJCAI. Seattle: Morgan Kaufmann Publishers, 2001. 467−472.

[9] Cimatti A, Roveri M, Bertoli P. Conformant planning via symbolic model checking and heuristic search. Artificial Intelligence, 2004,159(1-2):127−206.

[10] Cimatti A, Roveri M, Traverso P. Strong planning in non-deterministic domains via model checking. In: Simmons RG, Veloso MM, Smith S, eds. Proc. of the 4th Int'l Conf. on AI Planning Systems. Menlo Park: AAAI Press, 1998. 36−43.

[11] Cimatti A, Pistore M, Roveri M, Traverso P. Weak, strong, and strong cyclic planning via symbolic model checking. Artificial Intelligence, 2003,147(1-2):35−84.

[12] Cimatti A, Roveri M. Conformant planning via symbolic model checking. Artificial Intelligence Research, 2000,13(1):305−338.

[13] Bonet B, Geffner H. Planning with incomplete information as heuristic search in belief space. In: Chien S, Kambhampati S, Knoblock CA, eds. Proc. of the 5th Int'l Conf. on Artificial Intelligence Planning and Scheduling. Menlo Park: AAAI Press, 2000. 52−61.

[14] Hoffman J, Brafman R. Contingent planning via heuristic forward search with implicit belief states. In: Biundo S, Myers KL, Rajan K, eds. Proc. of the ICAPS. Menlo Park: AAAI Press, 2005. 71−80.

[15] Wu KH, Jiang YF. Planning with domain constraints based on model-checking. Journal of Software, 2004,15(11):1629−1640 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1629.htm

[16] Bertoli P, Cimatti A, Roveri M. Conditional planning under partial observability as heuristic-symbolic search in belief space. In: Cesta A, ed. Proc. of the ECP. Berlin, Heidelberg: Springer-Verlag, 2001. 379−384.

[17] Hoffman J, Braftman R. Conformant planning via heuristic forward search: A new approach. Artificial Intelligence, 2006,170(6-7): 507−541.

[18] Yin MH, Lin H, Sun JG, Wang JA. Extension or resolution: A novel approach for reasoning in possibilistic logic. In: Melin P, Castillo O, Gómez-Ramírez E, Kacprzyk J, Pedrycz W, eds. Proc. of the IFSA. Berlin, Heidelberg: Springer-Verlag, 2007. 354−362.

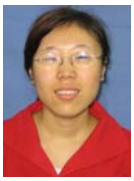
[19] Yin MH, Sun JG, Cai DB, Lu S. A novel framework for plan recognition: Planning graph as a basis. In: Veloso MM, ed. Proc. of the IJCAI WK on NRAC. Menlo Park: AAAI Press, 2007. 472−476.

[20] Yin MH, Lin H, Sun JG. Counting models using extension rules. In: Collins J, Faratin P, Parsons S, Rodríguez-Aguilar JA, Sadeh NM, Shehory O, Sklar E, eds. Proc. of the AAAI. Menlo Park: AAAI Press, 2007. 1390−1395.

[21] Domshlak C, Hoffmann J. Fast probabilistic planning through weighted model counting. In: Long D, Smith SF, Borrajo D, McCluskey L, eds. Proc. of the ICAPS. Menlo Park: AAAI Press, 2006. 1655−1664.

附中文参考文献:

[15] 吴康恒,姜云飞.基于模型检测的领域约束规划.软件学报,2004,15(11):1629−1640. http://www.jos.org.cn/1000-9825/15/1629.htm



周俊萍(1981－),女,吉林蛟河人,硕士,主要研究领域为智能规划,规划识别.

殷明浩(1979－),男,博士,助教,主要研究领域为智能规划,自动推理.

谷文祥(1947－),男,教授,博士生导师,主要研究领域为智能规划与规划识别,形式语言与自动机,模糊数学及其应用.

孙吉贵(1962－2008),男,教授,博士生导师, CCF 高级会员,主要研究领域为智能规划,自动推理.