第 39 卷 第 12 期
Vol.39 No.12

计 算 机 工 程
Computer Engineering

2013 年 12 月
December 2013

• 人工智能及识别技术 •　文章编号：1000—3428(2013)12—0162—05　文献标识码：A　中图分类号：TP18

不确定规划中一种观察信息高效约简算法

唐　杰[1]，文中华[1]，黄海平[1,2]，吴正成[1]

(1. 湘潭大学信息工程学院，湖南 湘潭 411105；2. 娄底职业技术学院，湖南 娄底 417000)

摘　要：在不确定规划中，可通过观察周围的信息来区分多个状态，但周围的观察信息较多，因此如何从大量的观察信息中筛选必须的信息非常重要。以往算法是在直接搜索过程中增加一些剪枝条件来达到优化的目的，存在一定的局限性。在对观察信息约简研究中，为提高搜索效率，设计一种高效的不确定规划中观察信息约简算法。该算法将规划问题转化为求解 0-1 矩阵的覆盖问题，使用数据结构十字链表来表示 0-1 矩阵，通过维护十字链表并采用启发式函数来加速求解一个最小观察变量集。实验结果表明，该算法不仅能够找最小观察变量集，而且运行速度超过同类算法。

关键词：不确定规划；观察信息约简；最小观察变量集；人工智能规划；十字链表；启发式搜索

An Efficient Algorithm for Observation Information Reduction in Nondeterministic Planning

TANG Jie[1], WEN Zhong-hua[1], HUANG Hai-ping[1,2], WU Zheng-cheng[1]

(1. College of Information Engineering, Xiangtan University, Xiangtan 411105, China;
2. Loudi Vocational and Technical College, Loudi 417000, China)

【Abstract】In nondeterministic planning, it is feasible to distinguish some states through observation information, however, observation information is so much that it is significant how to select useful information from plenty of observation information. The previous algorithms use direct search algorithm that adds some cutting condition to achieve optimized objective, however, these methods have certain limitations. In research to observation information reduction, this paper designs a new algorithm which improves search efficiency. It converts the problem into cover problem in 0-1 matrix through abstract problem model, and replaces the 0-1 matrix using orthogonal list data structure and can get a minimal set of observation variable through maintaining the orthogonal list and using heuristic function. Experimental result shows that this algorithm not only finds a minimal set of observation variable, but also runs more quickly than similar algorithm.

【Key words】nondeterministic planning; observation information reduction; minimal observation variable set; Artificial Intelligent(AI) planning; orthogonal list; heuristic search

DOI: 10.3969/j.issn.1000-3428.2013.12.035

1　概述

在现实生活中，会遇到很多的规划问题[1-2]，在规划的过程中会存在一些不确定性因素，而这些不确定因素可能会偏离所要达到的目标，也有可能接近所要达到的目标，正是因为这些问题的存在才会驱使人们对不确定规划做进一步研究[3-4]。

在不确定规划中，执行不确定动作会使得一个状态到达多个状态中的某一个，从而形成了一个状态的集合，通过观察信息就可以判断控制器在执行一个不确定动作后 agent 所在的状态，然而在执行过程中，并不需要所有观察信息，可以去除不必要的观察信息，达到减少开销的目的，因此，如何选取一个最小观察变量集非常重要。文献[5-6]给出了约简观察变量的方法，但都只能获得一个近似最小观察变量集。文献[5]是利用贪心算法进行求解。文献[6]是针对部分状态集进行求解，并且当问题存在一致性规划解时[7-8]，状态集不需要区分，这时求解的对象为空集。而在文献[9-10]中给出了求解最小观察变量集的方法。文献[9]设计的 ORSCP 算法是基于强循环规划求解最小观察变量集，并且没有给出优化算法。文献[10]中的 MOS 算法对 2 种情况进行了优化：第 1 种情况是假设 2 个观察变量 o_1 和 o_2 所能区分的状态对集合分别为 s_1 和 s_2，若 $s1 \subseteq s2$，则

基金项目：国家自然科学基金资助项目(61070232, 61272295)；湖南省重点学科建设基金资助项目(0812)

作者简介：唐　杰(1990－)，男，硕士研究生，主研方向：智能规划；文中华，副教授；黄海平、吴正成，硕士研究生

收稿日期：2012-10-25　**修回日期：**2012-12-29　**E-mail：**zhonghua@xtu.edu.cn

观察变量 o_1 是冗余的；第 2 种情况是当一个状态对只能被某一个观察变量区分时，那么这个观察变量必须选取。在随机生成的规划问题中这 2 个优化的效果并不明显。基于以上原因，本文采用十字链表数据结构来刻画问题的模型。在每次递归搜索的过程中，都使用启发式函数来求解从当前状态到目标状态至少需要多少观察变量并决定下一步是否需要进行，若不需要继续进行，则可以减少搜索的范围，若还需要进行，则使用贪心策略来决定下一个需要搜索的状态。

2　相关定义

不确定规划是智能规划的一个分支，其动作的执行存在不确定性，根据周围的环境是否能够被探测又可以分为完全可观察、部分可观察和完全不可观察的规划[11-12]。不确定规划的相关定义如下[13-14]:

定义 1 不确定规划领域 $D=<I, S, A, R>$是一个四元组，其中，S 是一个有限状态集合；$I \subseteq S$ 是初始状态集合；A 是有限动作集合，$R: S \times A \to 2^S$ 是一个状态转换函数。

R 用来刻画不确定性: 在状态 s 下执行动作 a 所得到的状态集合就是 $R(s, a)$。若 $R(s, a)$非空，则称动作 a 在状态 s 下是可执行的。在状态 s 下可执行的动作集合记作 $A(s)=\{a: \exists s' \in R(s, a)\}$。

定义 2 不确定规划问题 $P=<D, I, G>$是一个三元组，其中，D 是一个不确定规划领域；I 是一个初始状态集合；G 是一个目标状态集合，且 I、G 均不为空。

定义 3 设 S 是一个状态转换系统的状态集，O 是一个观察变量集，则 S 和 O 上的观察函数 $X: S \times O \to \{T, F\}$。

若一个观察变量 $o \in O$ 在一个状态 $s \in S$ 下确实存在，那么 $X(s, o)$为 T，否则为 F。

定义 4 一个规划 π_F 是一个(s, a)的集合，其中，$s \in S$，$a \in A(s)$，并且对于一个 s 最多只会存在一个 a 满足 $(s, a) \in \pi_F$。

定义 5 设 π 为不确定规划领域 D 的一个规划，则 π 在 D 上的执行结构为一个二元组的集合 $K=<Q, T>$，其中，$Q \subseteq S$，$T \subseteq S \times S$。一个最小的执行结构 K 应满足以下条件:

(1)若 $s \in I$，那么 $s \in Q$。

(2)若 $s \in Q$ 且存在某个 $(s, a) \in \pi$，$s' \in R(s, a)$，那么 $s' \in Q$ 且 $(s, s') \in T$。

状态 $s \in Q$ 是 K 的一个终止状态，当且仅当不存在 $s' \in Q$ 使得 $(s, s') \in T$。

例 1 图 1 展示了一个机器人导航的部署图。这个房间内共有 9 个位置，可以把它们看成 9 个子状态的集合 $S=\{s0, s1, \cdots, s8\}$。机器人可以在这些状态中移动，而作用在这些状态上的动作集合 $A=\{GoEast, GoSouth, GoWest, GoNorth\}$。如果在移动的方向上没有墙，对应的行为则可以执行，例如 $R(s1, GoSouth)=s4$。

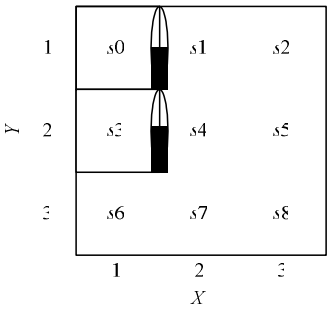


图 1　机器人导航部署图

假设除了 $R(s0, GoEast)=\{s1, s4\}$，$R(s3, GoEast)=\{s1, s4, s7\}$，$R(s4, GoEast)=\{s5, s7\}$是不确定的，其余的$(s, a)$都是确定的。那么设 $I=\{s0, s3\}$，$G=\{s6\}$，则 $\pi_F=\{<s0, GoEast>, <s1, GoEast>, <s2, GoSouth>, <s3, GoEast>, <s4, GoEast>, <s5, GoSouth>, <s7, GoWest>, <s8, GoWest>\}$是规划问题 $P=\{D, I, G\}$的一个规划解，执行结构 K 如图 2 所示。

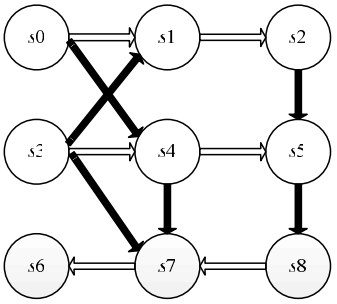


图 2　π_F 的执行结构

定义 6 设 $o \in O$，$s1, s2 \in S$，若 $X(o, s1) \neq X(o, s2)$，则称观察变量 o 可区分状态 $s1$ 和 $s2$。

定义 7 设 $E \subseteq S \times S$ 为一个状态对集合，一个最小观察集合 O_{obs} 满足 $\forall (s1, s2) \in E$，$\exists o \in O_{obs}$ 使得 $X(o, s1) \neq X(o, s2)$ 且 $|O_{obs}|$ 最小，$|O_{obs}|$ 为集合 O_{obs} 的元素个数。

3　不确定规划中观察信息约简

文献[5]最早提出约简观察信息，并且给出了一个求解近似最小观察集的方法。该方法首先求出一个需要区分的状态对集合 $E \subseteq S \times S$，然后利用贪心算法每次选取一个能够区分状态对最多的观察变量，直到所有的状态对被区分后才停止寻找，最后返回这个近似最小观察集。

最小观察集的求解很有可能会成为整个规划问题的瓶颈，因为最小观察变量集的求解是 NP 完全问题，若能够使用启发式方法快速地逼近并求出最优解，那么可以提升整个规划问题的执行效率。

下面将本文设计的算法称为 HSFMOS(Heuristic Search to Find Min Observation Set)算法。在算法中，首先找出所有需要区分的状态对集合 E，求解的方法是: 当状态的执行动作是不确定动作时，执行该动作的结果将是一个状态的集合，对于这个集合内的任意 2 个状态，假若它们下次执行

的动作不相同，那么这 2 个状态就是需要区分的状态对。求解状态对集合 E 后则对问题建立数学模型，设 V 是一个部分可观察集合，E 是需要区分的状态对集合，$n=|V|$，$m=|E|$，$\boldsymbol{H}$ 是一个 n 行 m 列的 0-1 矩阵，那么 3 个集合则分别为：$V=\{v_i\,|\,1\leqslant i\leqslant n\}$，$E=\{(a_i,b_i)\,|\,1\leqslant i\leqslant m\}$ 和 $\boldsymbol{H}=\{h_{ij}\,|\,1\leqslant i\leqslant n,1\leqslant j\leqslant m\}$。如果满足 $X(v_i,a_j)\neq X(v_i,b_j)$，则 $h_{ij}=1$，否则 $h_{ij}=0$。至此，已经将观察变量与状态对之间的关系用 0-1 矩阵表示出来，而求解最小观察集则转化为在 0-1 矩阵中寻找最小的行数使得每一列至少存在一个数字 1。

例 2 回顾例 1 中的机器人规划问题，再给其定义一个观察变量集 V={*WALLE*, *WALLS*, *WALLW*, *WALLN*}，分别代表东、南、西、北 4 个方向是否有墙。利用前面提到的方法对例 1 中的规划问题进行求解可以得到一个需要区分的状态对集合 E={(*s*1, *s*7), (*s*4, *s*7), (*s*2, *s*7), (*s*5, *s*7), (*s*5, *s*8)}，那么可以推算出 $\boldsymbol{H}$ 为：

$$\begin{array}{c} \\ WALLE \\ WALLS \\ WALLW \\ WALLN \end{array}\begin{array}{c} \begin{array}{ccccc} e1 & e2 & e3 & e4 & e5 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

*e*1 到 *e*5 顺序对应集合 E 中的每个元素，然后可以使用十字链表来表示上面的 0-1 矩阵，如图 3 所示。

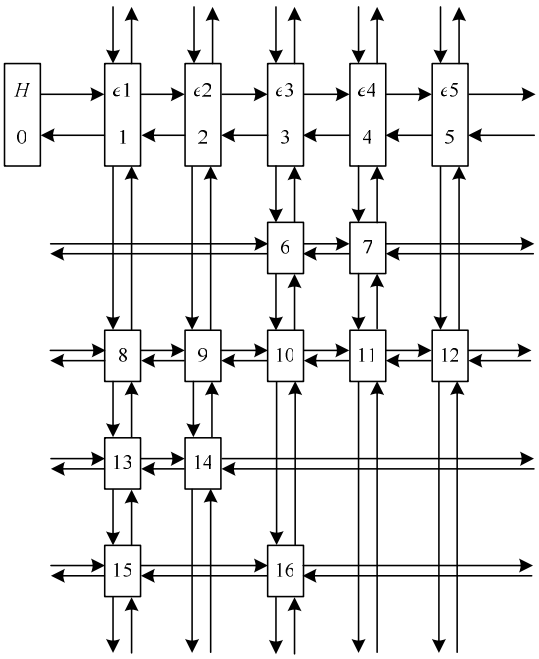


图 3 0-1 矩阵 H 的十字链表结构

图 3 中的数字代表节点所在数组的下标，最上面的一排是每列的头节点。

HSFMOS 算法的主流程如下：

```
1.function HSFMOS( π_F ,P,V,X)
2.fsps=FindStatePairs( π_F ,P);
3.ccls=CreateCrossLinks(fsps,V,X);
4.HeurSearch(0,ccls,ans=allObsNum);
5.return ans;
```

第 2 行的函数 FindStatePairs 用来找出所有需要区分的状态对，第 3 行的函数 CreateCrossLinks 根据状态对与观察变量之间的关系构造十字链表，这 2 个函数已经在前面使用文字和图片说明了其工作流程。第 4 行就是通过维护十字链表来搜索最优解，其中的 HeurSearch 函数是本文讨论的重点，HeurSearch 函数求解的是最小观察集中元素的数目，若要获得一个最小观察集，则可以再定义一个数组来存储这个集合。将 *ans* 初始化为观察变量总数后开始调用 HeurSearch 函数，HeurSearch 函数分为以下步骤进行处理：

(1)使用启发式函数估算出从当前状态到目标状态至少需要多少观察变量，将估算值与当前已经选择的观察变量数累加并与当前最优解 *ans* 进行比较，如果累加值超过 *ans* 则说明再往下搜索并不能得到一个比 *ans* 更优的解，于是可以停止往下搜索。

(2)若已经达到了目标状态，根据第(1)步可以知道当前这个解要比 *ans* 更优，于是更新最优解后停止往下搜索。

(3)在当前的十字链表结构中选取节点数最少的一列(状态对)，然后枚举这一列的每个节点对应的行(观察变量)，显然这里的每一行都能够区分被选择的这一列，接下来删除被选的这一行所能区分的所有列。由于这些列都已经可以被区分，因此在之后的搜索中并没有使用价值，这样下一步的搜索状态就是这个已经被修改的十字链表，在搜索返回之后需要恢复之前对十字链表的修改，即将被删除的列重新添加进来。

HeurSearch 函数的具体流程如下：

```
1. function HeurSearch(dep,ccls,ans)
2. if dep+HeurAccel(ccls) ⩾ ans then
3.     return;
4. end if
5. if ccls(0).r=0 then
6.     ans=dep;
7.     return;
8. end if
9. mc=INF_MAX;
10.for i from ccls(0).r until i=0 do
11.    if ccls(i).colnum<mc
12.        mc=ccls(i).colnum;
13.        nd=i;
14.    end if
15.    next i=ccls(i).r
16.end for
17.for i from ccls(nd).d until i=nd do
18.    Remove(i);
19.    for j from ccls(i).r until j=0 do
```

```
20.      remove(ccls,j);
21.      next j=ccls(j).r;
22.   end for
23.   HeurSearch(dep+1,ccls,ans);
24.   for j from ccls(i).l until j=0 do
25.      Resume(ccls,j);
26.      next j=ccls(j).l;
27.   end for
28.   resume(i);
29.   next i=ccls(i).d;
30.end for
31.end
```

在函数 HeurSearch 中，*ccls*(*i*).*l*、*ccls*(*i*).*r*、*ccls*(*i*).*u* 和 *ccls*(*i*).*d* 分别代表节点 *i* 在十字链表中的左、右、上和下节点，*ccls*(*i*).*colnum* 则代表节点 *i* 所在的列中节点的数目。

第 2 行用到了 HeurAccel()子函数估算还没有被区分的状态对至少需要多少观察变量。该函数使用了一个 *visit* 数组来记录状态对是否被区分。每次选择一个没有被区分的状态对，用 *visit* 数组标记已区分，然后将估算值加 1 并且将所有能够区分该状态的观察变量加入集合，最后删除掉与这些观察变量相关的所有列，即能够被这些观察变量区分的状态对，这里使用了标记 *visit* 数组来替换删除操作。例如程序第 1 次选择 *e*1 进行标记，那么观察变量 2、3、4 就会被选入观察变量集中，然后就删除掉观察变量 2、3、4 所关联的状态对，这里由于第 2 个观察变量与所有状态对都关联，因此这些状态对都会被标记为已区分，最后该函数返回的结果就是 1。

启发式函数的具体流程如下：

```
1. function HeurAccel(ccls)
2. for i from 0 to visit_size–1 do
3.   visit(i)=unvisited;
4. end for
5. ret=0;
6. while true do
7.    mc=–1;
8.    for c from ccls(0).r until c=0 do
9.       if visit(c)=unvisited ∧ (mc=–1 ∨
10.       ccls(mc).colnum >ccls(c).colnum) then
11.          mc=c;
12.      end if
13.   end for
14.   if mc=–1 then break;
15.   ret=ret+1;
16.   visit(mc)=1;
17.   for i from ccls(mc).d until i=mc do
18.      for j from ccls(i).r until j=i do
19.         visit(ccls(j).colindex)=visited;
20.         next j=ccls(j).r;
21.      end for
22.      next i=ccls(i).d;
23.   end for
24.   end if
25.   next c=ccls(c).r;
26.end for
27.return ret;
28.end
```

在 HeurSearch 函数中，9 行~16 行选取节点数最少的一列的优点是：假设某一列只有一个节点，那么可以肯定这个节点对应的行(观察变量)在后面的递归处理中是要选的，所以，一开始就选择这个观察变量能够减少后面的处理规模。该函数中还存在 2 个子函数来操作十字链表，一个子函数是 Remove(*ccls*, *col*)，用来移除节点 *col* 所在列的所有节点，而另一个子函数 Resume(*ccls*, *col*)则是恢复这一操作。

删除列操作流程如下：

```
1. function Remove(ccls,col)
2. i=ccls(col).d;
3. while i ≠ col
4.   ccls(ccls(i).r).l=ccls(i).l;
5.   ccls(ccls(i).l).r=ccls(i).r;
6.   i=ccls(i).d;
7. end while
8. end
```

恢复列操作流程如下：

```
1. function Resume(ccls,col)
2. i:=ccls(col).u;
3. while i ≠ col
4.   ccls(ccls(i).r).l=i;
5.   ccls(ccls(i).l).r=i;
6.   i=ccls(i).u;
7. end while
8. end
```

4 实验结果与分析

文献[9-10]中研究的都是最新的约简观察信息的方法。表 1 列出了文献[9]中的 ORSCP 算法、文献[10]中的 MOS 算法和 HSFMOS 算法在执行时间上的差异。对于每一次的比较都让 3 种算法执行了 50 组样例，然后记录执行 50 组样例的总时间。表 1 中第 1 行~第 6 行表示的是当观察变量数不变，状态对数递增时对实验结果的影响，观察变量数设置为 20 是因为它能比较好的体现状态对数对结果的影响，第 7 行~第 13 行表示的是当状态对数不变，观察变量数递增时对实验结果的影响，第 14 行~第 16 行表示的是当观察变量数和状态对数同时增加时对实验结果的影响。若执行时间大于 1 h 则表中用 TLE(Time Limit Exceed)来表示，

说明该算法执行该组数据的时间超过预期的上限时间。

表 1 算法执行时间比较

观察变量数	状态对数	执行时间/s		
		ORSCP 算法	MOS 算法	HSFMOS 算法
20	10	8.394	0.058	0.036
20	20	9.393	1.606	0.052
20	30	10.230	9.135	0.066
20	40	12.000	14.390	0.086
20	50	13.050	12.920	0.089
20	60	16.250	14.960	0.097
10	20	0.049	0.043	0.045
15	20	0.368	0.149	0.045
25	20	389.3	16.440	0.062
30	20	TLE	52.530	0.068
50	20	TLE	401.7	0.119
70	20	TLE	1 197	0.224
90	20	TLE	TLE	0.420
40	40	TLE	TLE	0.197
50	50	TLE	TLE	1.077
60	60	TLE	TLE	5.324

从表 1 可以看出，观察变量数对 ORSCP 算法和 MOS 算法的运行时间影响比较大，特别是 ORSCP 算法，能够处理的观察变量数非常小。而需要处理的问题正是对观察变量数进行约简，所以尽管 ORSCP 算法和 MOS 算法对状态对数不敏感，但它却不是所需要的。MOS 算法的运行时间与观察变量数和状态对数的差值也有很大的关系，当差值越大时，在搜索的过程中满足 MOS 算法的 2 个优化条件的可能性越大，运行时间增长的也会变慢，有时还会降低，这个可以从表 1 中的第 4 条和第 5 条数据看出来。HSFMOS 算法的运行时间与观察变量数和状态对数的总数有关，总数越大执行时间越长，但 HSFMOS 算法的时间增长速度明显低于前面 2 种算法，这说明 HSFMOS 算法采用的启发式方法达到了比较好的效果。

5 结束语

本文设计一个求解最小观察变量集的算法。该算法主要使用 2 种优化方法：(1)使用了十字链表数据结构，这种链式结构明显减少了删除和恢复数据的时间；(2)使用了启发式思想来搜索问题的解，这样能够减少许多不必要的搜索，从而快速找到问题的最优解。从实验结果可以看出，该算法在多种情况下都比 MOS 算法和 ORSCP 算法的效率高。今后还可以从以下 2 个方面进行研究：(1)对启发式函数进行进一步的优化或改进，从而降低搜索范围，提高效率。(2)本文使用的观察变量都具有相等的代价，还可以考虑向代价不相等的方向求解最小观察集。

参考文献

[1] Weld D S. Recent Advances in AI Planning[J]. AI Magazine, 1999, 20(2): 93-123.

[2] 吴康恒, 姜云飞. 基于模型检测的领域约束规划[J]. 软件学报, 2004, 15(11): 1629-1640.

[3] Bertoli P, Cimatti A, Roveri M, et al. Planning in Non-deterministic Domains Under Partial Observability via Symbolic Model Checking[C]//Proc. of the 17th International Joint Conference on Artificial Intelligence. Seattle, USA: [s. n.], 2001.

[4] Cimatti A, Roved M, Traverso P. Strong Planning in Non-deterministic Domains via Model Checking[C]//Proc. of the 4th International Conference on AI Planning Systems. Pittsburgh, USA: [s. n.], 1998.

[5] Huang Wei, Wen Zhonghua, Jiang Yunfei, et al. Observation Reduction for Strong Plans[C]//Proc. of the 20th International Joint Conference on Artificial Intelligence. Hyderabad, India: [s. n.], 2007.

[6] 周俊萍, 殷明浩, 谷文祥, 等. 部分可观察强规划中约减观察变量的研究[J]. 软件学报, 2009, 20(2): 290-304.

[7] Cimatti A, Roveri M. Conformant Planning via Symbolic Model Checking[J]. Artificial Intelligence Research, 2000, 13(1): 305-338.

[8] Cimatti A, Roveri M. Conformant Planning via Model Checking and Heuristic Search[J]. Artificial Intelligence, 2004, 159(1): 127-206.

[9] 常 青, 文中华, 胡雨隆, 等. 强循环规划的观察信息约简[J]. 计算机工程与应用, 2012, 48(2): 148-150.

[10] 饶东宁, 蒋志华, 姜云飞, 等. 对不确定规划中观察约简的进一步研究[J]. 软件学报, 2009, 20(5): 1254-1268.

[11] Cimatti A, Pistore M, Roveri M, et al. Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking[J]. Artificial Intelligence, 2003, 47(1): 35-84.

[12] Bertoli P, Cimatti A, Roveri M, et al. Strong Planning Under Partial Observability[J]. Artificial Intelligence, 2006, 170(4): 337-384.

[13] Pistore M, Traverso P. Planning as Model Checking for Extended Goals in Nondeterministic Domains[C]//Proc. of the 17th International Joint Conference on Artificial Intelligence. Seattle, USA: [s. n.], 2001.

[14] Fu Jicheng, Ng V, Yen I L, et al. Simple and Fast Strong Cyclic Planning for Fully-observable Nondeterministic Planning Problems[C]//Proc. of the 22th International Joint Conference on Artificial Intelligence. Barcelona, Spain: [s. n.], 2011.

编辑 索书志