分层法求最小权值强规划解

伍小辉[1] 文中华[2,3] 李 洋[1] 劳佳琪[1]
(湘潭大学信息工程学院 湘潭 411105)[1] (湖南工程学院计算机与通信学院 湘潭 411104)[2]
(湘潭大学智能计算与信息处理教育部重点实验室 湘潭 411105)[3]

摘 要 在不确定规划领域中,以往对强规划解的研究侧重于解本身,很少考虑不确定转移系统执行动作所需的代价;而已有的研究最小权值强规划解的算法效率不高。针对这一问题,引入模型检测的强规划分层方法,设计了一种快速求解最小权值强规划解的算法。该算法首先将不确定规划问题中的状态进行强规划分层,然后利用分层信息反向搜索最小权值强规划解;且在搜索的过程中,根据算法策略,实时更新所需搜索层数的上界和下界,从而避免了大量的无用搜索,提高了搜索效率。实验表明:所设计的算法能快速求解出最小权值强规划解,求解效率比已有的直接求解最小权值强规划解的算法高;且分层数和动作数越大,优势越明显。

关键词 不确定规划,最小权值强规划解,模型检测,强规划分层方法

中图法分类号 TP18 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.2.047

Solving Minimal Cost Strong Planning Solution by Hierarchical Algorithm

WU Xiao-hui[1] WEN Zhong-hua[2,3] LI Yang[1] LAO Jia-qi[1]
(College of Information Engineering, Xiangtan University, Xiangtan 411105, China)[1]
(Department of Computer & Communication, Hunan Institute of Engineering, Xiangtan 411104, China)[2]
(Key Laboratory of Intelligent Computing & Information Processing, Ministry of Education, Xiangtan University, Xiangtan 411105, China)[3]

Abstract In nondeterministic planning areas, previous studies on the strong planning solution focused on the solution itself, with little regard for the required cost of nondeterministic transfer system performing an action, and the algorithms' efficiency which exists is not high. Aiming at this problem, we introduced a strong planning hierarchical method in model-checking, designed an algorithm to solve the minimal cost strong planning solution quickly. Firstly, this algorithm uses strong planning hierarchical method to get hierarchical states of nondeterministic problem, and then it reversely searches the minimal cost strong planning solution by using the hierarchical information. In the search process, according to the algorithm strategy, the upper and lower bounds of required searching layer are real-time updated to avoid a lot of useless search, improving search efficiency. Experimental results show that this algorithm can not only solve the minimal cost strong planning solution quickly and precisely, but also run more efficient than existing algorithms. And the greater the number of layers and the number of actions, the more obvious the advantages.

Keywords Nondeterministic planning, Minimal cost strong planning solution, Model-checking, Strong planning hierarchical method

1 引言

智能规划[1]是人工智能研究领域的一个重要分支,而不确定规划[2]又是智能规划中的热点问题,具有突出的理论和应用价值。Silva等人在文献[3]中就利用不确定规划器来自动生成交互式的故事情节;Patrizi等人在文献[4]中则利用强循环规划器来解决公平情况下的线性时序逻辑综合问题;另外,Fu等人在文献[5]中提出了一种在全可观察条件下快速求解强循环规划解的简单算法;而文献[6]也是基于不确定规划模型来解决实际问题的。

规划解和观察信息约简等内容是不确定规划研究的重点。Cimatti等人在文献[7,8]中给出了有关强规划解、弱规划解和强循环规划解的完整定义和说明,并尝试使用模型检测[9,10]中的规划方法求解规划解;Bertoli等人在文献[11]中探讨了在部分可观察下的强规划问题;唐杰等人在文献[12]中提出了不确定可逆规划问题并设计了求解其强循环规划解的算法;此外,黄巍等人在文献[13]中提出了部分可观察下的观察信息约简问题。

在客观世界中,由于任何动作的执行都是需要代价的,因此可以在不确定规划中引入数值规划的思想,对执行的每个动作赋予一个权值。从而就引出了一个新的问题:带权值的不确定规划问题。在这个新问题中,如何求解动作权值之和

到稿日期:2014-03-06 返修日期:2014-07-22 本文受国家自然科学基金(61272295,61105039,61202398),湘潭大学智能计算与信息处理教育部重点实验室,湖南省重点学科建设项目(0812),湖南省教育厅一般项目(12C0399)资助。

伍小辉(1988—),男,硕士生,主要研究方向为智能规划,E-mail:510280596@qq.com;文中华(1966—),男,博士,教授,主要研究方向为智能规划;李 洋(1991—),男,硕士生,主要研究方向为智能规划;劳佳琪(1990—),男,硕士生,主要研究方向为智能规划。

最小的规划解就成为了迫切需要解决的问题，因此，求解强、弱、强循环规划解的目标也就转换成求解最小权值强、弱、强循环规划解了。

针对这一问题，本文对如何求解带权值的不确定规划问题的最小权值强规划解[14]展开研究，提出了分层法求解最小权值强规划解的算法。该算法不但可以准确地求解出最小权值强规划解；而且通过引入模型检测的强规划分层方法，可以避免大量的无效搜索，从而提高了搜索效率。实验验证了所设计的算法比用文献[14]的直接求解最小权值强规划解的算法的效率高。

2 相关定义

定义 1(带权值的不确定规划领域) 带权值的不确定规划领域 $\sum=\langle S,A,\gamma\rangle$，其中，$S$ 是一个有限状态集，A 是一个带权值的有限动作集，$\gamma.\ S\times A\to 2^S$ 是状态转换函数。

γ 用来表示不确定性：$\gamma=(s,a)$ 表示状态 s 执行动作 a 所得到的状态集合；若 $\gamma=(s,a)$ 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 $A(s)=\{a:\exists s'\in\gamma(s,a)\}$；其中，称 s' 是 s 可达的，称 (s,a) 为状态动作序偶。每个动作的权值用 $cost_a(a\in A)$ 表示。

定义 2(带权值的不确定规划问题) 带权值的不确定规划问题 $P=\langle\sum,S_0,S_g\rangle$，其中 $\sum$ 是不确定规划领域，$S_0\subseteq S$ 是初始状态集合，$S_g\subseteq S$ 是目标状态集合。

定义 3(带权值的不确定规划的执行结构) 带权值的不确定规划的执行结构 $K=\langle Q,T\rangle$，其中，$Q\subseteq S$ 和 $T\subseteq S\times S$ 是满足以下条件的最小集合：

(1)若 $s\in S$，那么 $s\in Q$。

(2)若 $s\in Q$ 且存在某个动作 a 使得 $(s,a)\in\pi$，那么对所有的 s'(其中 $s'\in\gamma(s,a)$)都有 $s'\in Q$ 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状态 $s'\in Q$，使得 $(s,s')\in T$。K 是一个有向图，其中，Q 是在执行规划解时能够到达的所有状态集合，T 表示所有可能的状态转移。显然，K 的终止状态代表规划执行的终止，这里用 $S_{terminal(K)}$ 表示执行结构 K 的终止状态集。

定义 4(不确定规划的规划解) 设 $\sum=\langle S,A,\gamma\rangle$ 是一个不确定规划领域，$P=\langle\sum,S_0,S_g\rangle$ 是 $\sum$ 上的一个不确定规划问题，π 是不确定规划领域 $\sum$ 中的一个状态动作序偶表，$K=\langle Q,T\rangle$ 是 π 从 S 导出的执行结构。

(1)π 是 P 的弱规划解当且仅当 $(\forall s\in S_0)(\exists s'\in S_g\cap S_{terminal(K)})$，其中 s' 是 s 可达的。

(2)π 是 P 的强循环规划解当且仅当 $S_{terminal(K)}\subseteq S_g$，且 $(\forall s\in Q)(\exists s'\in S_{terminal(K)})$，其中 s' 是 s 可达的。

(3)π 是 P 的强规划解当且仅当 K 是无环的，且 $S_{terminal(K)}\subseteq S_g$。

若 P 有强规划解，则称可以从初始状态强到达目标状态。

定义 5(不确定规划的最小权值强规划解) 设 $\sum=\langle S,A,\gamma\rangle$ 是一个不确定规划领域，$P=\langle\sum,S_0,S_g\rangle$ 是 $\sum$ 上的一个不确定规划问题，C_π 是求解得到强规划解 π 所需的动作权值之和。则 π 是 P 的最小权值强规划解当且仅当 π 是 P 的强规划解，且 $\forall\pi',C_{\pi'}\geqslant C_\pi$。

3 算法实现及复杂度分析

由最小权值的定义(定义 5)可知，执行最小权值强规划解的状态动作序偶集合 π_{min}，不但可以保证系统能从初始状态到达目标状态，而且所需动作的权值之和最小。

3.1 算法思想

设带权值的不确定规划问题 P 的状态集 St 中含有 n 个状态 $s_1,s_2,\cdots,s_n$；$cost_a(a\in Act(s_i))$ 表示 $\gamma(s_i,a)$ 所需的代价，即动作权值；其中 $Act(s_i)$ 是从状态 s_i 出发的动作集；$dist_i(1\leqslant i\leqslant n)$ 用于保存从状态 s_i 出发到达目标状态的强规划解的最小代价值(即强规划解中的最小的动作权值之和)；$sAct_i(1\leqslant i\leqslant n)$ 用于保存以状态 s_i 为初始状态的最小权值强规划解；$sSet$ 为已经求得的到达目标状态集具有最小权值强规划解的状态集合；$S_i(1\leqslant i\leqslant n)$ 是分层之后第 i 层的状态集合；$Top_i(1\leqslant i\leqslant n)$ 记录第 i 层状态集合 $S_i(1\leqslant i\leqslant n)$ 中的状态个数。

算法首先根据模型检测的强规划分层方法对状态集 St 中的状态进行分层，得到 $S_i(1\leqslant i\leqslant n)$。然后根据求得的分层信息，从目标状态开始，反向搜索最小权值强规划解；且在搜索的过程中，根据算法策略，实时更新所需搜索层数的上界 up 和下界 $down$，这样就只需遍历 $S_i(down\leqslant i\leqslant up)$ 中的状态，而无需搜索其余状态，从而避免了大量的无用搜索，提高了搜索效率。其中 $down$ 是分层状态集合 $S_i(1\leqslant i\leqslant maxLayer)$ 中还有状态没有求得最小权值强规划解的最小层数，up 是分层状态集合 $S_i(1\leqslant i\leqslant maxLayer)$ 中已经有状态求得最小权值强规划解的最大层数再加 1。

定理 1 搜索最小权值强规划解时，只需搜索分层状态集合中 S_{down} 到 S_{up} 中的状态。

证明：(1)因为 $down$ 是分层状态集合 $S_i(1\leqslant i\leqslant maxLayer)$ 中还有状态没有求得最小权值强规划解的最小层数，也就是说，S_1 到 S_{down-1} 中的所有状态到目标状态的最小权值强规划解都已经求出，显然无需再搜索这些集合中的状态。

(2)由强规划分层定义可知，第 $i+1$ 层中的状态只能强到达第 i 层，而不能强到达第 $i-1,i-2,\cdots,1$ 层。

而文献[14]在第 3.1 节(算法的思想)第 3 段证明了以状态 s_i 为初始状态的最小权值强规划解只有两种情况，即：要么 $\exists a,\gamma(s_i,a)\subseteq S_g,a\in Act(s_i)$；要么从 s_i 出发，中间只经过 $sSet$ 中的顶点而最后到达目标状态集 S_g。

由前面的定义可知，up 是分层状态集合 $S_i(1\leqslant i\leqslant maxLayer)$ 中已经有状态求得最小权值强规划解的最大层数再加 1。所以层数大于 up 的状态集合中的任意状态 s_i，它执行任意动作之后所达到的状态都不包含于目标状态集 S_g 中，也不包含在 $sSet$ 中，所以它无法求出到达目标状态的最小权值强规划解。因此，层数大于 up 的状态集合中的状态也无需搜索。即：$\forall s_i,\forall a,\gamma(s_i,a)\not\subset\{S_g\cup sSet\}$，其中 $a\in Act(s_i)$，$s_i\in\{S_{up+1},\cdots,S_{maxLayer}\}$。

由(1)、(2)可以证明，在搜索最小权值强规划解时，只需搜索分层状态集合中 S_{down} 到 S_{up} 中的状态。

3.2 算法实现

强规划分层和初始化函数如下：

```
1. Function InitAndHierarchical(S0,Sg)
2. S1=Sg;
3. layer=1;
4. while(NewSA≠∅∧S0⊄(Slayer∪Slayer-1∪…∪S1))
5.    layer=layer+1;
6.    NewSA=STRONGNEWSA(Slayer∪Slayer-1∪…∪S1);
7.    Slayer=GETSTATES(NewSA);
```

```
8.   Top_layer = CountNewSANum(NewSA);
9. end while
10. if(S_0 ⊄ (S_layer ∪ S_layer-1 ∪ … ∪ S_1))
11.   return false;
12. end if
13. for each i (1≤i≤Top[2])
14.   if(∃a,a∈Act(s_i) ∧ s_i∈S_2)
15.     dist[s_i]=cost_a;
16.     sAct[s_i]=sAct[s_i] ∪ {a};
17.   end if
18. end for
19. return S,dist,sAct,Top;
```

第 4－9 行是对状态进行强规划分层处理。其中第 4 行是判断上一次分层是否成功，如果成功，且初始状态集 S_0 中还有状态未包含于已有的分层之中，则执行 while 循环；否则，循环结束。第 6 行的函数 STRONGNEWSA(St)＝{(s, a)：$s\notin St, \gamma(s,a)\neq\emptyset$}是从尚未包含于 St 集合的状态中找出能够强到达 St 的状态，并存储在集合 $NewSA$ 中。第 7 行的函数 GETSTATES(π)＝{s：(s,a)$\in\pi$}是将所求得的新一层的状态集赋值给 S_{layer}；而第 8 行的函数 CountNewSANum($NewSA$)是求出该层的状态个数，并用 Top_{layer} 来保存。

第 10－12 行是判断是否有强规划解。如果初始状态集 S_0 中的所有状态都包含于强规划分层中，说明强规划解存在，则继续执行后面的代码；否则，说明强规划解不存在，返回 false。

第 13－18 行是初始化 $dist$ 和 $sAct$。因为开始时所有状态到达目标状态的最小权值强规划解都没有求出，所以此时 $down=2, up=1+1=2$，因此只需初始化第 2 层即可。其中 $dist[s_i](s_i\in S_2)$用于保存从状态 s_i 出发到达目标状态的强规划解的最小代价值，$sAct[s_i](s_i\in S_2)$用于保存从状态 s_i 出发到达目标状态的最小权值强规划解。

最小权值强规划解搜索函数如下：

```
1. function MinCostStrongSearch(S_0,S_g)
2. InitAndHierarchical(S_0,S_g);
3. sSet=S_g;
4. for each i (1≤i≤n)
5.   if( S_0⊆sSet)
6.     break;
7.   end if
8.   if(∃j,s_j∉sSet ∧ dist_j=min{dist_m1,dist_m2,…,dist_mn},{s_m1,s_m2,
     …,s_mn}⊆{S_down,S_down+1,…,S_up})
9.     x=j;
10.    sSet=sSet ∪ {s_x};
11.    if(s_x∈S_up) up++;
12.    if(∀k,s_k∈S_down ∧ s_k∈sSet) down++;
13.  end if
14.  for each j (down≤j≤up)
15.    for each k (1≤k≤Top[j])
16.      if(∃a,a∈Act(s_k) ∧ s_k∉sSet ∧ γ(s_k,a)={s_x,s_m1,s_m2,…,
         s_mn},{s_x,s_m1,s_m2,…,s_mn}⊆sSet)
17.        minCost=dist_x+dist_m1+dist_m2+…+dist_mn;
18.        if(∀s_v,s_w,∃a_y,a_y∈sAcvt_v ∧ a_y∈sAcvt_v,{v,w}⊆{x,
           m_1,m_2,…,m_n})
19.          minCost=minCost−cost[a_v];
20.        end if
21.        if(minCost<dist_k)
22.          dist_k=minCost;
23.        sAct_k=sAct_x ∪ sAct_m1 ∪ sAct_m2 ∪ … ∪ sAct_mn;
24.        end if
25.      end if
26.    end for
27.  end for
28. end for
29. return dist,sAct;
```

第 2 行是调用函数 InitAndHierarchical(S_0, S_g)来对状态进行强规划分层，并初始化 $dist$ 和 $sAct$。

第 4－28 行是求解最小权值强规划解。

其中，第 5－7 行是循环终止条件，即：初始状态集 S_0 中的所有状态到达目标状态的最小权值强规划解都已经求出。

第 8－13 行是从已经求出强规划解的状态中找出权值最小的那个状态 x，将 x 加入 $sSet$，并更新 $down$ 和 up 的值。其中第 8 行中的 if 语句是判断的条件，用于找出上一次搜索到的到达 $sSet$ 集合具有最小权值的状态 x；第 10 行是将状态 x 加入 $sSet$ 集合；第 11、12 行是分别根据之前设定的规则来更新 up 和 $down$ 的值。

第 14－27 行是在 x 加入 $sSet$ 后，更新 $down$ 层到 up 层中的状态到达目标状态的最小权值强规划解。其中从第 14 行的循环条件可以看出，该算法只需搜索第 $down$ 层到第 up 层之间的状态，而无需搜索所有的状态，可以减少搜索的时间代价。第 15，16 行是对尚未包含于 $sSet$ 集合中但是能强到达 $sSet$ 集合的状态，执行第 17 行的代码，找出它到达 $sSet$ 所需的最小权值；然后执行第 18－20 行的代码，即判断状态 s_k 当前所求得的最小权值强规划解中是否有重复的状态动作序偶，如果状态 s_k 的最小权值强规划解中有重复的状态动作序偶，则去掉那些重复的状态动作序偶；最后执行第 21－24 行的代码，即判断当前求出的强规划解的最小权值是否小于原来求得的最小权值，如果小于，则更新状态 s_k 到达目标状态的最小权值强规划解。

执行完第 14－27 行的循环之后，$down$ 层到 up 层中的状态到达目标状态的最小权值强规划解就都分别求出来了。此时重新返回执行第 4 行的循环，并在下一轮的循环中，在第 8－13 行会找出 $down$ 层到 up 层中所有尚未加入 $sSet$ 集合的状态中权值最小的状态 x 并将其加入 $sSet$，然后再执行 14－27 行的循环。如此重复，直至所有状态的最小权值强规划解都求出并加入 $sSet$ 集合。

3.3 算法时间复杂度分析

该算法分为两个部分：(1)强规划分层及初始化；(2)搜索初始状态的最小权值强规划解。设动作集大小为 m，状态集大小为 n。

如果出现极端情况，即所有状态最终只有两个分层，则分层函数第一次搜索只需 n 次，第二次搜索小于 n 次，此时具有最好时间复杂度，且复杂度为 $O(n)$；而当出现 n 个层次时，因为每个层次只有一个状态，但是搜索判断时，剩下的状态每次都要判断一次，判断次数依次为 $n, n-1, \cdots, 1$，所以最坏时间复杂度为 $O(n+(n-1)+\cdots+2+1)=O((n+1)*n/2)=O(n^2)$。

执行最小权值强规划解搜索函数时，若只有两个分层，且第 2 层的状态的动作集总大小为 k，则此时它具有最好的时

间复杂度，且时间复杂度为 $O(k)(1\leqslant k\leqslant m)$。而当只有两个分层，目标状态只有 1 个，且它的动作集为空时，那么 *down* 和 *up* 变量对减少状态搜索约束的作用实际上就完全没有了，此时具有最坏的时间复杂度 $O(m_{n-1}+m_{n-2}+\cdots+m_1)$，其中，$m_i$ 是第 i 次搜索时，剩余状态的动作集的总大小，且 $m_1\leqslant m, m_i<m(1<i\leqslant n-1)$，所以最坏时间复杂度为 $O(n*m)$。而分层的层次大于 2 时，由于 *down* 和 *up* 变量的影响，它们的时间复杂度是介于上述二者之间的。

设 $x=\max\{m,n\}$，则该算法的最坏时间复杂度为 $O(n*x)$。

4 算法示例分析

图 1 是一个带权值的不确定规划领域 $\Sigma=\langle S,A,\gamma\rangle$，$P=\langle\Sigma,S_0,S_g\rangle$ 是 Σ 上的一个规划问题。其中，$S_0=\{s_1\}$ 是初始状态集合，$S_g=\{s_{11}\}$ 是目标状态集合。规划问题 P 是在规划领域 Σ 上求出从初始状态集合 S_0 出发到达目标状态集合 S_g 的最小权值强规划解。

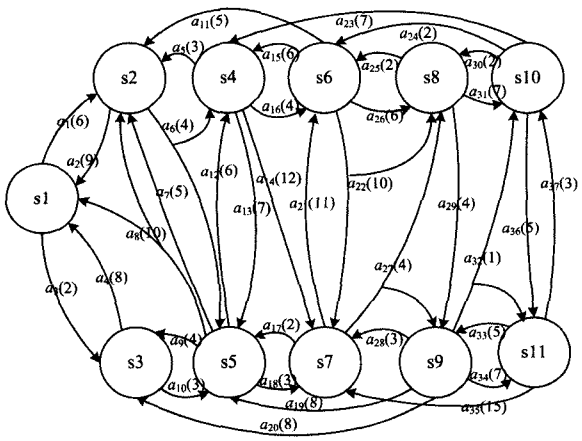

图 1 带权值的不确定规划领域

算法首先根据模型检测的强规划分层方法对状态集 *St* 中的状态进行分层，得到分层信息 *S*，其中 $S_1=\{s_{11}\}$，$S_2=\{s_{10},s_9\}$，$S_3=\{s_8\}$，$S_4=\{s_7,s_6\}$，$S_5=\{s_5,s_4\}$，$S_6=\{s_3,s_2\}$，$S_7=\{s_1\}$。然后初始化 *dist* 和 *sAct*，可以得到 $dist_{10}=cost[a_{36}]=5$，$dist_9=cost[a_{34}]=7$。

然后执行最小权值强规划解搜索函数。

第 1 次搜索：$sSet=\{s_{11}\}$，此时 $down=2$，$up=2$。因为 $dist_{10}=\min\{S_2\}=\min\{dist_9,dist_{10}\}=\min\{7,5\}=5$。更新 *down*、*up* 和 *sSet*，此时 $down=2$，$up=3$，$sSet=\{s_{11},s_{10}\}$。再更新 $\{S_{down},S_{down+1},\cdots,S_{up}\}$ 中尚未找到最小权值强规划解的状态的 *dist* 值。所以 $dist_9=\min\{dist_9,dist_{10}+cost[a_{32}]\}=\min\{7,6\}=6$；$dist_8=cost[a_{31}]+dist_{10}=7+5=12$。

第 2 次搜索：$sSet=\{s_{11},s_{10}\}$，此时 $down=2$，$up=3$。因为 $dist_9=\min\{S_2,S_3\}=\min\{dist_9,dist_8\}=\{6,12\}=6$。更新 *down*、*up* 和 *sSet*，此时 $down=3$，$up=3$，$sSet=\{s_{11},s_{10},s_9\}$。再更新 $\{S_{down},S_{down+1},\cdots,S_{up}\}$ 中尚未找到最小权值强规划解的状态的 *dist* 值。所以 $dist_8=\min\{dist_8,dist_9+cost[a_{32}]\}=\{12,6+4\}=\min\{12,10\}=10$。

然后再循环，执行第 3 次搜索、第 4 次搜索等，直到初始状态集合中的所有状态都找到最小权值强规划解，则停止搜索。

通过上述搜索，最终得到从初始状态集 S_0 出发到达目标状态集 S_g 的最小权值强规划解为：$\pi=\{(s_1,a_3),(s_3,a_{10}),(s_5,a_{18}),(s_7,a_{27}),(s_8,a_{29}),(s_9,a_{32}),(s_{10},a_{36})\}$；最小权值为 22。

5 算法实验分析

本文实验环境为：Window7＋Pertinum® Dual-Core CPU T4400 @ 2.2GHz＋2.00GB 内存＋Code::Blocks 10.05。

由于文献[14]首先提出最小权值强规划解概念，且目前暂无其他文献直接研究该问题，因此本文先设计实验与该文算法进行横向比较。文献[14]的算法在实验中用“算法 1”表示，本文算法用“算法 2”表示。两种算法在状态数为 50，70，100，300，500，700，1000，1300，1500，2000，3000，5000 的不确定规划领域下，分别运行 100 组实验数据的平均运行时间比较如表 1 所列。

表 1 求解最小权值强规划解的运行时间比较

状态数	算法 1	算法 2
50	0.058	0.007
70	0.1	0.015
100	0.213	0.072
300	2.657	0.771
500	8.43	2.49
700	18.601	5.22
1000	24.736	8.013
1300	42.936	14.448
1500	58.978	19.643
2000	107.96	36.442
3000	262.953	86.133
5000	904.402	267.88

通过表 1 可以看出，本文的算法在求解最小权值强规划解时，运行时间明显优于文献[14]的算法，从而验证了在引入分层策略后，通过限定搜索的上界 *up* 和下界 *down*，可以减少大量不必要的搜索，从而减少程序的运行时间。

但是，表 1 只是横向比较了分层策略和上下界的限定对减少搜索的作用，却无法反映权值的引入对规划解搜索的影响。为了探究这一问题，本文通过引入反向搜索强规划解和正向搜索强规划解两个算法，新增加了一个纵向比较的实验。其中本文的算法用“算法 1”表示，反向搜索算法用“算法 2”表示，正向搜索算法用“算法 3”表示。3 种算法在状态数为 50，70，100，300，500，700，1000，1300，1500，2000，3000，5000 的不确定规划领域下，分别运行 100 组实验数据的平均运行时间比较如表 2 所列。

表 2 搜索强规划解的运行时间比较

状态数	算法 1	算法 2	算法 3
50	0.007	0	0
70	0.015	0	0.001
100	0.072	0.015	0.005
300	0.771	0.016	0.013
500	2.49	0.598	0.432
700	5.22	1.314	1.244
1000	8.013	2.742	2.613
1300	14.448	6.012	5.936
1500	19.643	8.402	6.768
2000	36.442	51.782	14.246
3000	86.133	67.504	23.279
5000	267.88	303.05	46.023

表 2 中的“算法 1”和“算法 3”都使用了分层策略。从表 2 中可以看出，“算法 1”搜索最小权值强规划解所需的时间比

"算法 3"单纯求规划解的时间要长很多。这验证了在引入权值后，为了求得最优值，会大量增加搜索的时间代价。

而表 2 中的"算法 1"和"算法 2"则体现了分层策略和权值两个因素的综合影响。当状态数少于 2000 时，"算法 1"搜索规划解的时间要比"算法 2"长很多；而当状态数大于 2000 时，两种算法的运行时间差不多。结合该现象和算法本身，通过分析可知：在引入权值后，搜索最小权值规划解时，需要考虑所有的强规划解，从而增加了搜索代价；而分层策略的引入虽然可以减少很多无用搜索，但是分层本身也是需要代价的，所以在状态数比较少时，"算法 1"消耗的时间要明显比"算法 2"长；但是，当状态数较多时，分层所需的代价就处于次要地位了，它减少搜索的优势明显体现出来，所以此时"算法 1"的搜索时间和"算法 2"差不多，甚至有时还要略少于"算法 2"。

通过上面的两个横向和纵向的实验分析，我们可以得到这样的结论：1) 通过引入分层策略和设置搜索的上下界，确实可以减少搜索代价；2) 引入权值后，最优搜索需要考虑所有规划解的耗费情况，它的搜索代价比简单的规划解搜索代价要大。

结束语 针对带权值的不确定规划问题，本文设计了一种求解最小权值强规划解的算法。该算法首先使用模型检测的强规划分层方法对状态集进行强规划分层，然后反向搜索各状态到达目标状态的最小权值强规划解，直到求出初始状态集中所有的状态的最小权值强规划解，停止搜索。实验结果表明，使用本文设计的分层法求强规划解算法，可以求出最小权值强规划解，从而验证了算法的有效性；而且该算法可以减少很多无用搜索，效率比以往的算法更高。

今后还可以从以下几个方面进行研究：

(1)结合最新的强规划解搜索算法，进一步优化求解最小权值强规划解的算法。

(2)将本文设计的算法进行改进，用于求解最小权值弱规划解和最小权值强循环规划解。

(3)本文研究的带权值不确定规划问题的研究对象主要是单个 Agent，以后可以将其扩展至多 Agent 领域。

参 考 文 献

[1] Ghallab M, Nau D, Traverso P. Automated Planning Theory and Practice[M]. Massachusetts: Morgan Kaufmann Publishers, 2004:1101-1132

[2] 丁德路，姜云飞. 智能规划及其应用的研究[J]. 计算机科学，2002,29(2):100-103

[3] Da Silva F A G, Ciarlini A E M, Siqueira S W M. Nondeterministic planning for generating interactive plots[M]. Berlin: Springer Berlin Heidelberg, 2010:133-143

[4] Patrizi F, Lipovetzky N, Geffner H. Fair LTL synthesis for non-deterministic systems using strong cyclic planners[C]// Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. USA: AAAI Press, 2013:2343-2349

[5] Fu J, Bastani F B, et al. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems[C]// IJCAI Proceedings-International Joint Conference on Artificial Intelligence. 2011:1949-1954

[6] 薛晗. 不确定规划的群体智能计算[D]. 长沙：国防科学技术大学，2009

[7] Cimatti A, Roveri M, Traverso P. Strong planning in nondeterministic domains via model checking[C]// Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems (AIPS' 98). USA: Carnegie Mellon University, 1998:36-43

[8] Cimatti A, Pistore M, Roveri M, et al. Weak, strong, and strong cyclic planning via symbolic model checking[J]. Artificial Intelligence, 2003, 147(1/2):35-84

[9] Cimatti A, Roveri M. Conformant planning via model checking andheuristic search[J]. Artificial Intelligence, 2004, 159(1/2): 127 -206

[10] 文中华，黄巍，刘任任，等. 模型检测规划中的状态分层方法[J]. 软件学报，2009,20(4):858-869

[11] Bertoli P, Cimatti A, Roveri M, et al. Strong planning under partial observability[J]. Artificial Intelligence, 2006, 170(4/5): 337-384

[12] 唐杰，文中华，汪泉，等. 不确定可逆规划的强循环规划解[J]. 计算机研究与发展，2013,50(9):1970-1980

[13] Huang Wei, Wen Zhong-hua, Jiang Yun-fei, et al. Observation reduction for strong plans[C]// Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07). India: IJCAI, 2007:1930-1935

[14] 陈建林，文中华，马丽丽，等. 一种求解最小权值强规划的方法[J]. 计算机工程，2011,37(17):167-171

(上接第 209 页)

[11] 史开泉. P-集合，逆 P-集合与信息智能融合-过滤识别[J]. 计算机科学，2012,39(4):1-13

[12] Varshney P K. Multisensor data fusion [J]. Journal of Electronics Computer Engineering, 1997, 9(6):245-253

[13] 史开泉. P-推理与信息的 P-推理发现-辨识[J]. 计算机科学，2011,38(7):1-9

[14] 于秀清，徐风生. P-规律推理与未知规律发现-应用[J]. 山东大学学报：理学版，2012,47(1):110-115

[15] 林宏康，范成贤，史开泉. 倒向 P-推理与属性剩余发现-应用[J]. 计算机科学，2011,38(10):189-198

[16] 李豫颖，林宏康，史开泉. 数据离散区间特征与数据发现-应用[J]. 系统工程与电子技术，2011,33(10):2258-2262

[17] Lin Hong-kang, Li Yu-ying. P-sets and its P-separation theorems[J]. An International Journal Advances in Systems Science and Applications, 2010, 10(2):209-215

[18] 赵树理，吴松丽，史开泉. 内 P-推理信息恢复与属性潜藏推理发现[J]. 计算机科学，2013,40(4):209-213

[19] Fan Cheng-xian, Lin Hong-kang. P-sets and the reasoning-identification of disaster information[J]. International Journal of Convergence Information Technology, 2012, 7(1):337-345

[20] Lin Hong-kang, Fan Cheng-xian. The dual form of P-reasoning and identification of unknown attribute[J]. International Journal of Digital Content Technology and its Applications, 2012, 6(1): 121-131

[21] 史开泉. P-信息规律智能融合与软信息图像智能生成[J]. 山东大学学报：理学版，2014,49(4):1-17