分层法求强循环规划解

汪 泉 文中华 伍 选 唐 杰
(湘潭大学信息工程学院 湘潭 411105)

摘 要 设计了一种求解强循环规划问题的状态分层算法。从目标状态开始,首先进行强规划分层,然后对剩余状态进行弱规划分层,并记录相应信息,最后用该信息作启发因子,在弱规划分层结果中搜索强循环规划分层。分层结束后利用分层时记录的信息可以直接得到强循环规划解。所设计的算法在求解状态动作较多的强循环规划问题时有较高的效率;且当强规划解存在时,求解效率更高,并能保证得到质量更优的强循环规划解——强规划解。实验表明,所设计的算法能够以较少的重复搜索得到强循环规划解,求解效率比反向搜索高。

关键词 强循环规划,状态分层,不确定规划,智能规划
中图法分类号 TP18 **文献标识码** A

Hierarchical Algorithm Solve Strong Cycle Planning

WANG Quan WEN Zhong-hua WU Xuan TANG Jie
(College of Information Engineering, Xiangtan University, Xiangtan 411105, China)

Abstract A hierarchical algorithm was designed to solve strong cycle planning. Hierarchical algorithm is start with the target state, first, uses strong planning hierarchies, second, uses the weak planning hierarchies with the remaining states, and records the appropriate information, finally uses that information as a heuristic factor to search a strong cycle planning hierarchy in the result of weak planning hierarchies. After hierarchical states, information recorded can be used to get strong cycle planning solution directly. When larger state action pair exists, designed algorithm has high efficiency. When strong planning solution exists, it can owe better efficient, and can ensure a better strong cycle planning solution—strong planning solution obtained. Experiments show that designed algorithm can get strong cycle planning solution by fewer repeat searches, is better than the backward search by high efficiency.

Keywords Strong cycle planning, Hierarchical states, Nondeterministic planning, Intelligent planning

1 前言

不确定规划能够更好地解决现实中众多带有不确定性的问题[1],是当前智能规划研究的热点。模型检测规划是一种重要的规划方法[2,3],能够用来求解带有不确定性的规划问题[4,5]。模型检测规划根据规划解分为强规划、强循环规划和弱规划3类[6]。强循环规划是其中最复杂[7,8]的一类规划问题,仍有待进一步研究。

当前求强循环规划解主要有以下两类方法:从目标状态开始的反向搜索算法[9,10];在弱规划解的基础上,通过启发式搜索等进行扩充的算法[11,12]。用反向搜索算法求强循环规划解时,由于循环的存在,需要多次反复搜索。采用弱规划解扩充法时,由于许多弱规划解并不能扩充为强循环规划解,且扩充过程中的无效搜索会非常多,因此需要根据具体情况设计好的启发式来加速搜索过程。

通过状态分层[13]能够将大量不必要的状态动作对去掉,从而较大程度地减小了问题规模,提高了问题求解效率。

强规划解是强循环规划解的特例,比一般的强循环规划解优越,并且其求解过程更为简单。将强规划求解方法以适当方式统一到强循环规划的求解算法中,可以提高强循环规划问题的求解效率;并能在强规划解存在时,保证得到质量更优的强循环规划解——强规划解。

本文在反向搜索基础上,将分层思想运用于强循环规划的求解。设计了一种强循环规划分层算法,其通过在强循环规划分层中调用强规划分层方法,以及利用分层过程中记录的可达关系等启发式信息,提高了强循环规划分层的搜索速度,并且在分层结束后能直接得到强循环规划解。

2 相关定义

定义1 一个规划领域是一个不确定的状态转移系统 $\Sigma=(S,A,\gamma)$[1],其中,S 是有限状态集,A 是有限动作集,$\gamma: S\times A\rightarrow 2^S$ 是状态转移函数。

在状态 s 下可执行动作的集合记为 $A(s)=\{a: \exists s'\in \gamma(s,a)\}$,并称 (s,a) 为状态动作序偶。

定义2 $D=(N,H)$ 为超图,其中,N 为结点集,H 为超弧集。超弧是一个有序对,其第1个元素为 N 中的一个结

到稿日期:2013-01-24 返修日期:2013-04-07 本文受国家自然科学基金项目(61070232,61272295)资助。
汪 泉(1989—),男,硕士生,主要研究方向为云计算、智能规划,E-mail:1026336374@qq.com;文中华(1966—),男,博士,副教授,主要研究方向为智能规划;伍 选(1988—),男,硕士生,主要研究方向为智能规划;唐 杰(1990—),男,硕士生,主要研究方向为智能规划。

点，第2个元素为 N 的一个子集[13]。

定义3 设 $\Sigma=(S,A,\gamma)$ 是一个不确定的有限状态转移系统，称 $\Sigma D=(S,E)$ 为 $\Sigma=(S,A,\gamma)$ 的超图[13]。

如果 $\exists a,\gamma(s,a)=\{s_1\}$，则 $(s,\{s_1\})\in E$；如果 $\exists a,\gamma(s,a)=\{s_1,s_2,\cdots,s_k\}(k\geqslant 2)$，则 $(s,\{s_1,s_2,\cdots,s_k\})\in E$。$S$ 中的元素为超图 ΣD 的顶点，对应于 D 中的 N；E 中的元素为超图 ΣD 的超弧，对应于 D 中的 H。

如果 $(s,\{s_1,s_2,\cdots,s_k\})\in E$，则对于任意的 $s_i\in\{s_1,s_2,\cdots,s_k\}$，$(s,s_i)$ 为超图 ΣD 的超弧边，顶点 s 为超弧边 (s,s_i) 的起点，顶点 s_i 为超弧边 (s,s_i) 的终点。

一个不确定的状态转移系统唯一确定一个超图，反之亦然。

定义4 规划领域 Σ 下的规划问题 P 是一个三元组 (Σ,S_0,S_g)，其中，$S_0\subseteq S$ 是初始状态集合，$S_g\subseteq S$ 是目标状态集合[6]。

定义5 设 $\Sigma=(S,A,\gamma)$ 是一个规划领域，$P=(\Sigma,S_0,S_g)$ 是 Σ 上的一个规划问题，π 是规划领域 $\Sigma=(S,A,\gamma)$ 中的一个状态动作序偶表。K 为从某个初始状态开始执行 π，直至到达目标状态或者状态转移停止时结束可能经过状态的转移路径。

定义6 若存在 $\exists K$：K 从初始状态开始目标状态结束，则称 π 为规划问题的弱规划解。

定义7 若任意 $\forall K$：K 从初始状态开始目标状态结束，则称 π 为规划问题的强循环规划解。

定义8 若任意 $\forall K$：K 从初始状态开始目标状态结束，且路径中不存在返回已经过状态的环，则称 π 为规划问题的强规划解。

由定义可知，强规划解一定是强循环规划解，强循环规划解一定是弱规划解。

3 状态分层及强循环规划求解方法

在强循环规划分层算法中使用了强规划分层和弱规划分层。利用强规划分层和弱规划分层的信息大幅减少强循环规划分层搜索的重复次数，并且充分利用分层中已经记录的信息，使分层结束后可以直接得到强循环规划解。

3.1 3种状态分层要求达到的效果

强规划分层要求保证：对于任意被分层状态 s，只经过其前面各层状态（目标状态集为最前面层状态），就能得到以 $\{s\}$ 为初始状态集、原目标状态集 S_g 为目标状态集的规划问题的强规划解。

弱规划分层要求保证：对于任意被分层状态 s，只经过其前面各层状态，就能得到以 $\{s\}$ 为初始状态集、原目标状态集 S_g 为目标状态集的规划问题的弱规划解。

强循环规划分层要求保证：对于任意被分层状态 s，只经过被分层状态，就能得到以 $\{s\}$ 为初始状态集、原目标状态集 S_g 为目标状态集的规划问题的强循环规划解。

3.2 强循环规划分层的处理过程

强循环规划分层算法中首先调用强规划分层算法，以强规划分层的方式处理掉大量状态，这些状态在强循环规划的后续搜索过程中将不再涉及。这样既避免了在这些状态中使用一般强循环规划分层方式需要的反复搜索过程，也为后续的复杂搜索过程缩减了问题规模，而且利用强规划分层可以判断规划问题是否存在强规划解。当强规划解存在时，可以直接结束强循环规划分层，并利用当前的分层信息得到质量更好的强循环规划解——强规划解。当强规划解不存在时，强规划分层结果可以作为强循环规划分层的最底层部分。

算法在强规划分层后采用弱规划分层方法处理剩余状态，为强循环规划分层后续的反复搜索过程排除掉大量不可能状态，进一步缩小问题规模，并为后续的过程记录了大量有用信息，减少了之后搜索过程的重复次数。

强循环规划分层的后继过程（强循环分层）中，对于当前分层中的每个状态 s_i，如果 s_i 所选定动作 a_i 可能的结果状态在当前分层之外，则说明该状态动作对 (s_i,a_i) 不能构成强循环规划解。首先尝试选择 s_i 的其它动作，若存在可能结果都落在当前分层内的动作 a_k，则选择该动作，否则删除状态 s，直到当前分层中的每个状态 s_i，其选定动作 a_i 的可能结果都在当前分层中，算法结束。

强循环分层中，由于每删除一个状态都可能造成：一些状态动作对，由所有结果状态都落在当前分层内变为存在结果状态落于当前分层外。所以这一过程是一个复杂的反复过程。为了提高效率，减少重复搜索的次数，并不搜索当前分层中的所有状态，而是利用弱规划分层中记录的信息 *StatesFromS0ToOut* 作启发因子，只搜索 *StatesFromS0ToOut* 中的状态，即从初始状态可达，且最终可能到达分层之外的状态，并在搜索过程更新 *StatesFromS0ToOut* 信息。

3.3 强循环规划解求取

在强循环规划分层中，已经记录了分层中各状态所执行的动作，所以可直接得到强循环规划解：强循环分层中的所有节点及其执行的动作。分层过程记录了各状态是否从初始状态集可达的信息。所以只需要通过简单的判断，选择从初始状态可达的状态集，就能得到不包含非必要状态动作对的精确强循环规划解。

4 算法实现

以下为强规划分层、弱规划分层、强循环规划分层的具体实现。为适用强循环规划的分层和求解要求，对强规划分层和弱规划分层做了相应修改。

4.1 强规划分层

强规划分层算法如下：

```
1. Function strongHierarchies(S0,Sg)
2.   i=1 //层号
3.   layer[i]=Sg
4.   i++
5.   foreach(s in layer[i])
6.     foreach(arc can to s)
7.       delete(arc)
8.       if(arc is last arc of hostA(arc))
9.         layer[i]=layer[i]+(holdS(arc),holdA(arc))
10. i++
11. if(layer[i]== ∅)
12.   return {fail,layer}
```

```
13. else if(S0 in layer)
14.   return {OK,layer}
```

其中，*holdS* 为引出超弧边 *arc* 的状态，*holdA* 为相应状态中引出超弧边 *arc* 的具体动作。

第 7 行，删除可以到达本层状态的超弧边。第 8、9 行，若某动作对应的超弧边已经全部被删除，则说明相应状态执行该动作，可能的结果状态一定为当前已分层状态，将其加入到下一层状态。

4.2 弱规划分层

在强规划分层结束后，若已分层状态未包含全部初始状态，则将已分层状态集合作为目标状态，对未分层状态进行弱规划分层。

弱规划分层算法如下：

```
1. Function weakHierarchies(S0,Sg)
2.   i=maxStrongLayerNo+1   //层号
3.   layer[i]=Sg
4.   while(layer[i]!=∅)
5.     i++
6.     foreach(s in layer[i])
7.       foreach(arc can to s)
8.         sa=((holdS(arc),holdA(arc))
9.         layer[i]=layer[i]+sa
10.        if(layer[i]==∅)
11.          if(S0⊆layer)
12.            recordStatesFromS0()
13.            return {success,layer}
14.          else
15.            return {fail,layer}
```

为求弱规划解而采用的弱规划分层算法，在已分层状态中包含了所有初始状态后就可以停止。但在求强循环规划解时，要对弱规划分层进行相应修改，使其在已包含所有初始节点时不结束，而是等到没有节点可以加入到分层时才结束，以保证所有强循环规划层状态都被包含在弱规划分层中。

为便于强循环规划分层快速地从弱规划分层中删除无关状态，并选择正确的动作，采用 *recordStatesFromS0*()函数间接记录已分层状态中从初始状态可达的状态 *stateFromS0*，以及 *stateFromS0* 中最终有可能到达外部状态(未被分层的状态)的状态 *StatesFromS0ToOut*。

recordStatesFromS0()实现方式为：在各状态下记录可以从初始状态到达并且不经过其他状态就能直接到达当前状态的所有状态。

为表述方便，将 *recordStatesFromS0*()的调用写在第 12 行。具体实现时，该函数的功能可以适当提前调用，并在新状态加入时根据需要调用，以便于在特殊情况下提前结束整个分层过程。因为当初始状态集合中的所有状态都已经被分层以后，如果 *StatesFromS0ToOut* 为空，则说明当前的弱规划分层已经是强循环规划分层，可以直接结束。但该记录功能一般不在 S_0 中状态全部被添加到分层之前调用，否则可能造成大量重复无效的记录操作，增加运行时间。

4.3 强循环分层

在弱规划分层结束后，若已分层状态未包含全部初始状态，则说明规划问题没有弱规划解，也就没有强循环规划解；若已包含了全部初始状态，便可以从弱规划分层中删除无关状态，并为相关状态选择正确动作得到强循环规划分层。

强循环分层算法如下：

```
1. Function strongCycleHierarchies(weakResult)
2.   while(StatesFromS0ToOut!= ∅)
3.     state=selectAstate(statesFromS0ToOut)
4.     newAct=findOtherActToLayer()
5.     if(newAct!=∅)
6.       stateUsingNewAct(state,newAct)
7.     else
8.       delFromLayer(state,weakResult)
9.       if(state ∈ S0)
10.          return fail
11.        recordStatesFromS0ToOut()
12. SAs={(s,a):s ∈ stateFromS0,s using a}
13. return {success,SAs}
```

第 11 行的 *recordStatesFromS0ToOut*()函数主要用于记录信息更新。具体实现时，可根据情况在第 7 行选用新动作的操作中调用，以及在第 9 行删除状态的操作中调用。

5 算法示例分析

例 1　图 1 是一个不确定状态转移系统 $\Sigma=(S,A,\gamma)$ 的超图，$P=(\Sigma,S_0,S_g)$ 是 Σ 上的一个规划问题，其中 $S_0=\{s_1\}$ 是初始状态集合，$S_g=\{s_{12}\}$ 是目标状态集合，规划问题 P 是在 Σ 上求从 S_0 到 S_g 的强循环规划解。

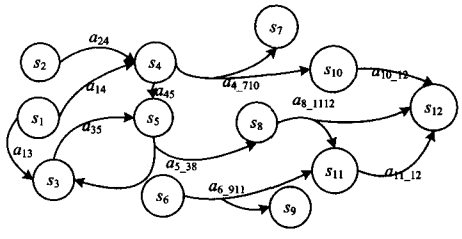

图 1　一个不确定状态转移系统的超图

首先采用强规划分层从 $\{s_{12}\}$ 开始得到第 1 层 $\{s_{12}\}$、第 2 层 $\{s_{10},s_{11}\}$ 和第 3 层 $\{s_8\}$，此时其它状态都不能确定到达已分层状态，强规划分层结束。

然后以这 3 层为初始状态进行弱规划分层，得到第 4 层 $\{s_4,s_5,s_6\}$ 和第 5 层 $\{s_2,s_1,s_3\}$，弱规划分层完成。其中第 4 层状态 s_4 是从 s_{10} 状态搜索得到的，其选定的动作一定是 a_{4_710}。

接着用 *recordStatesFromS0*()函数记录和更新可达关系等信息。

最后利用记录的信息在弱规划分层中搜索强循环规划分层和强循环规划解。

(1)假设 s_1 选定的动作是 a_{13}。从 s_1 开始可达的状态为 $s_3,s_5,s_8,s_{11},s_{12}$，所执行的动作不可能到达未分层状态。得到强循环规划解$(s_1,a_{13}),(s_3,a_{35}),(s_5,a_{5_38}),(s_8,a_{8_1112}),(s_{11},a_{11_12})$。

(2)假设 s_1 选定的动作是 a_{14}。因为 s_4 选定的动作是 a_{4_710}，从 s_1 开始可达的状态为 s_4,s_7,s_{10},s_{12}，其中 s_4 可能到达未分层状态 s_7。删除状态动作对(s_4,a_{4_710})，并为 s_4 选择新动

作 a_{45}。s_1 可达的状态更新为 $s_4, s_5, s_3, s_8, s_{11}, s_{12}$，不会到达未分层状态。得到强循环规划解$(s_1, a_{14}), (s_4, a_{45}), (s_5, a_{5_38}), (s_3, a_{35}), (s_8, a_{8_1112}), (s_{11}, a_{11_12})$。

(3)假设 s_1 选定的动作是 a_{14}，且图中 s_4 的动作 a_{45} 不存在，则 s_4 中不存在所有可能结果都落在分层内的动作。从分层中删除 s_4，此时 s_1 可达的状态为外部状态 s_4，为 s_1 选择新动作 a_{13}，得到与(1)相同的可达关系和强循环规划解。

6 实验分析

以下为采用分层搜索和反向搜索求强循环规划解的实验结果比较。其中表1为规划问题没有强规划解时求强循环规划解的实验结果对比，表2为规划问题有强规划解时求强循环规划解的实验结果对比。实验环境均为 Windows7＋Pentium® D CPU 3.00Hz＋2.0GB 内存＋VC＋＋Express 2012。实验数据为随机生成。两个算法使用的数据输入输出过程相同，故其运行时间没有包括在内。

表1 无强规划解时的运行时间对比

状态数	分层法搜索时间/ms	反向搜索时间/ms
50	0.154957	0.146766
80	0.208885	0.207178
200	0.724612	0.489105
300	0.954317	0.883665
400	1.17412	1.13624
500	1.25741	1.10381
800	2.30627	2.21889
2000	6.19316	5.87165
3000	8.72641	12.4662
4000	11.7075	17.9132
5000	15.9698	19.0017
8000	26.0837	33.8588

表2 有强规划解时的运行时间对比

状态数	分层法搜索时间/ms	反向搜索时间/ms
50	0.0873767	0.141646
80	0.115023	0.247112
200	0.390123	0.505488
300	0.481254	0.887761
400	0.627679	1.33693
500	0.831444	1.98645
800	1.28915	2.24005
2000	2.99538	6.86999
3000	4.44904	8.85611
4000	5.98599	15.222
5000	7.90111	22.7179
8000	12.3112	34.7193

从表1中可以发现，当状态数很少时，分层法搜索由于要记录较多信息，运行时间比反向搜索长；但当状态很多时，分层方法的优势就突显出来了，运行速度比反向搜索要快。

从表2中可以发现，当规划问题存在强规划解时，分层法的运行速度比反向搜索有更显著提高，而且状态越多，速度提升越大。

结束语 本文通过整合强规划分层和弱规划分层，设计了一种用于求解强循环规划问题的分层算法。充分利用强规划和弱规划分层的优势，以及分层过程中记录的信息，提高了强循环规划问题的求解效率及解的质量。

本文的进一步工作主要有：

(1)利用分层中记录的信息，进一步改进搜索策略；

(2)将本文分层思想运用于规划领域其他问题的求解，如：规划领域可达关系研究等。

参 考 文 献

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

[2] Cimatti A, Roved M, Traverso P. Strong planning in nondeterministic domains via model checking[C]//Proceedings of the 4th International Conference on AI Planning Systems. 1998: 36-43

[3] Cimatti A, Roveri M. Conformant planning via symbolic model checking[J]. Journal of Artificial Intelligence Research, 2000, 13: 305-338

[4] Bertoli P, Cimatti A, Roveri M, et al. Planning in nondeterministic domains under partial observability via symbolic model checking[C]//Proceedings of the 17th Int'l Joint Conf. on Artificial Intelligence(IJCAI 2001). Morgan Kaufmann Publishers, 2001: 473-478

[5] Pistore M, Traverso P. Planning as model checking for extended goals in nondeterministic domains[C]//Proceedings of the 17th International Joint Conference on Artificial Intelligence(IJCAI' 2001). 2001: 479-484

[6] Cimatti A, Pistore M, Roveri M, et al. Weak, strong, and strong cyclic planning via symbolic model checking[J]. Artificial Intelligence, 2003, 47(1/2): 35-84

[7] Daniele M, Traverso P, Vardi M Y. Strong cyclic planning revisited[C]// Proceedings of the 5th European Conf. on Planning (ECP'99). London: Springer-Verlag, 1999: 35-48

[8] Bertoli P, Cimatti A, Roveri M, et al. Strong planning under partial observability[J]. Artificial Intelligence, 2006, 170: 337-384

[9] Rintanen J. Backward plan construction under partial observability[C]//Proceedings of the 6th Int'l Conf. on Artificial Intelligence Planning Systems (AIPS' 2002). AAAI Press, 2002: 173-182

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

[11] Kuter U, Nau D, Reisner E, et al. Using classical planner to solve nondeterministic[C]// Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'2008). AAAI Press, 2008: 190-197

[12] Fu Ji-cheng, Ng V, Bastani F B, et al. Simple and Fast Strong Planning For Fully-Observable Nondeterministic Planning Problems[C]//Proceedings of the 22th Int'l Joint Conf. on Artificial Intelligence(IJCAI 2011). Morgan Kaufmann Publishers, 2011: 473-478

[13] 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