基于有序爬山法的前向启发式搜索规划

梁瑞仕[1]，姜云飞[2]，杨会志[1]

(1. 电子科技大学中山学院计算机学院 广东 中山 528402; 2. 中山大学信息科学与技术学院 广州 510275)

【摘要】根据快速前向启发式搜索规划器FF中放宽规划图和有利动作之间的关系，定义了状态适用度函数的概念，可对后继扩展状态的启发式估值进行快速近似的比较。在此基础上，结合增强型爬山法搜索后继状态的贪婪选择机制，提出了一种改进的局部搜索算法——有序爬山法，即根据状态适应度函数对所有待扩展的后继状态进行排序，并加入到扩展优先队列。在启发式估值阶段，适应度高的状态将被优先计算评估，有利于更快地发现较优状态，从而减少调用启发式估值程序的次数。通过在国际规划大赛基准测试领域的实验结果表明，该方法减少了搜索节点的数目和搜索时间，有效地提高了启发式搜索效率，而计算状态适应度及对状态进行排序的时间消耗几乎可以忽略，因此整体规划性能比FF有显著的提升。

关 键 词 智能规划; 启发式搜索; 有序爬山法; 状态适应度函数

中图分类号 TP18 **文献标志码** A doi:10.3969/j.issn.1001-0548.2013.03.028

Forward Heuristic Search Planning Based on Ordered Hill Climbing Algorithm

LIANG Rui-shi[1], JIANG Yun-fei[2], and YANG Hui-zhi[1]

(1. School of Computer, Zhongshan Institute, University of Electronic Science and Technology of China Zhongshan Guangdong 528402;
2. School of Information Science & Technology, Sun Yat-Sen University Guangzhou 510275)

Abstract According to the relationship between relaxed planning graph and helpful actions in forward heuristic search planner FF, the notion of state adaptive function is defined, which is used to fast compare heuristic evaluations for expanded successor states approximately. Integrating with greedy selection mechanism in enforced hill climbing search, we propose an improved local search algorithm named ordered hill climbing search algorithm based on state adaptive function. The core idea of our algorithm is to order all expanded successor states according to their state adaptive functions, and then insert them into an expanded priority queen. In heuristic evaluation stage, a state with higher adaptive value will be computed earlier, as a result, better state can be found earlier and the frequency of calling heuristic evaluation procedure will be cut down. Experiments in Depots and FreeCell domains of IPC show that the proposed algorithm reduces search nodes and search time significantly and therefore improves the search efficiency effectively.

Key words AI planning; heuristic search; ordered hill climbing algorithm; state adaptive function

启发式搜索是在智能规划(AI planning)领域中应用的最广泛和成功的技术之一[1]，是当前智能规划研究领域的热点。由于采用启发式搜索技术，智能规划在近十几年来取得了长足的进步和发展，各种新的启发式技术及相应的高性能规划器不断涌现[2]，如HSP[3]、FF[4]、Fast Downward[5]、LAMA[6]等，推动了智能规划社区的发展。

快速前向规划器(fast forward，FF)[4, 7]是一种高效的启发式搜索规划器，在国际智能规划大赛IPC-2、IPC-3中获得最佳性能奖[2]。它的高效主要基于以下几项关键技术：1) 构建放宽的规划图(relaxed planning graph)，用于计算状态的启发式估值；2) 采用增强型爬山法(enforced hill climbing)，作为快速前向启发式搜索算法；3) 定义有利动作(helpful actions)，作为有效的剪枝策略。FF的成功为启发式搜索规划方法的研究提供了一个良好的范例，基于放宽规划启发式和前向局部搜索的思想已经被广泛用于求解各类复杂规划问题。

在启发式搜索过程中，对每个扩展状态调用启发式估值程序通常是搜索过程中最耗时的部分，后继扩展状态的顺序将对搜索效率产生明显的影响[8]。理想情况下，如果在搜索过程中，每次都能

收稿日期：2011 – 08 – 15; 修回日期：2011 – 12 – 20
基金项目：国家自然科学基金(60773201, 60970042)；广东省自然科学基金(S2012040011123)
作者简介：梁瑞仕(1982 –)，男，博士，主要从事智能规划、启发式搜索等方面的研究.

将最好的节点放在最前面，那么局部搜索将仅需沿着一条深度优先的路径到达目标节点。放宽规划图作为一种有效的启发式估值方法，其中隐含了丰富的领域知识。受此启发，本文从放宽规划图和有利动作之间的关系出发，定义了状态适应度函数的概念，用于对后续扩展状态的启发式估值进行快速近似的比较。在此基础上，利用增强型爬山法搜索后继节点的贪婪选择机制，提出了一种改进的局部搜索算法(有序爬山法)，可快速地引导启发式搜索过程，提高启发式搜索效率。实验结果表明了该算法的有效性。

本文介绍了相关背景知识即STRIPS规划任务描述和放宽规划图启发式，讨论了FF的启发式搜索机制和增强型爬山法，介绍了本文的核心思想状态适应度函数和有序爬山法。

1　背景知识

本文介绍了STRIPS规划任务描述和放宽规划图启发式，相关知识可参考文献[4, 9-10]。

1.1　STRIPS规划任务描述

命题STRIPS规划语言采用一阶逻辑表示，P是逻辑命题的有限集合；状态s是一组逻辑命题的集合，表示该状态下命题取值为真；动作a是一个三元组a=(pre(a)，add(a)，del(a))，分别表示动作的前提、增加表(正效果)、删除表(负效果)，其中，pre(a)、add(a)、del(a)均为逻辑命题集合。如果pre(a)⊆s，即动作a可在状态s下执行，则称动作a可应用于状态s，动作执行的结果为exec(s, a)= $s\cup$add(a)−del(a)；否则，exec(s, a) = undefined。状态s下所有可应用的动作集合为$A(s)=\{a\in A \mid \text{pre}(a)\subseteq s\}$，状态$s$的所有后继状态集合为$N(s)=\{\text{exec}(s, a) \mid a\in A(s)\}$。如果状态$s'\in N(s)$，则称$s'$是$s$的一个邻居或邻居状态。

STRIPS规划任务是一个四元组T=(P, A, I, G)。其中，P为逻辑命题的有限集合，A为所有动作的有限集合，I、G分别表示规划任务的初始状态和目标状态。若存在一个动作序列$\pi=<a_1,a_2,\cdots,a_n>$，且$G\subseteq$exec(I, π)，则称π是规划任务T的一个解，其中，exec(I, π) = exec (exec (I, a_1)，$<a_2,a_3,\cdots,a_n>$)。

1.2　放宽规划图启发式

给定一个STRIPS规划任务T=(P, A, s, G)，它的放宽是T^+=(P, A^+, s, G)。T^+的放宽规划图(RPG)是一个命题集合和放宽动作集合交替出现的层次图结构，分别标记为P_0, A_0,⋯, P_i, A_i,⋯, A_{t-1}, P_t，其中有：

$$P_i := \begin{cases} s & i=0 \\ P_{i-1}\cup\{\text{add}(a) \mid \text{pre}(a)\subseteq P_{i-1}\} & i>0 \end{cases}$$

$$A_i := \{a\in A^+ \mid \text{pre}(a)\subseteq P_i \quad i\geqslant 0\}$$

式中，t是放宽规划图的最后一层，有2种情况$P_t=P_{t-1}$，表明图已经到达不动点或者$G\subseteq P_t$，说明此时已经达到规化目标，可以提取放宽的规划解。实际上，FF的放宽规划图是Graphplan中的规划图的简化[11]。

对于规划问题T=(P, A, s, G)，FF对状态s的启发式估值包括：1) 构建放宽的规划图；2) 从放宽的规划图中反向提取放宽规划解。如果放宽问题存在规划解(即$G\subseteq P_t$)，则经过解提取过程，会得到2个集合(序列)：1) 记录命题层P_i($1\leqslant i\leqslant t$)出现的子目标的集合G_i($1\leqslant i\leqslant t$)；2) 记录从动作层A_i($0\leqslant i\leqslant t-1$)所提取的满足相关子目标的解动作的集合$O_i$($0\leqslant i\leqslant t-1$)。$G_i$和$O_i$组成放宽规划解图，最终得到的动作序列$<O_0, O_1,O_2,\cdots,O_{t-1}>$是放宽的规划解。

FF采用放宽规划解的动作数目总和作为状态s到目标状态的启发式距离估值，即 $h(s)=\sum_{i=0}^{t-1}|O_i|$。若无解，则$h(s)=\infty$。除此之外，FF还利用放宽规划图计算状态$s$的有利动作集合，记为：

$$\text{HP}(s)=\{a\in A \mid \text{pre}(a)\subseteq s \wedge \text{add}(a)\cap G_1\neq\varnothing\}$$

2　基于有序爬山法的前向启发式搜索规划

2.1　FF的增强型爬山法

FF采用增强型爬山法(enforced hill climbing，EHC)执行局部搜索，该方法结合了局部搜索和系统搜索的优点。它执行如下的宽度优先搜索：从当前状态节点s出发，先将它所有的直接后继节点(子状态)放入一个(先进先出)扩展优先队列中，然后从队列首部开始执行宽度优先搜索，计算队列中每一个状态s'的启发式估值h，如果遇到$h(s')<h(s)$，则搜索成功，返回s'，从s'开始进入下一个局部搜索迭代过程，直到$h(s)=0$；否则，当$h(s')\geqslant h(s)$时，则将s'从队列中移除，再将s'的所有子状态放到队列尾部；然后继续从队列首部取出下一个状态执行搜索，直到找到一个较好的状态或者遇到局部极小值。增强型爬山法的搜索过程如图1所示。

EHC的特点如下：在搜索过程中，采用宽度优先搜索，一旦找到一个满足要求的状态节点，则本次搜索结束，所有其他状态的节点都不再计算而全部舍弃，这是一种典型的贪婪选择机制；否则会将该状态的子节点加入到队列中，这是与普通爬山法

不同之处。普通爬山法只搜索当前状态的子节点层(第1层)，不考虑再次一级的子节点，即不会搜索到更深的层次。而EHC在直接子节点中无法找到满足要求的状态节点时，会继续第2、3…层的搜索，直至成功或者全部状态都搜索完毕。搜索过程是最耗时的过程，因为每次评估一个节点都要调用一个放宽图规划程序求解它的启发式估值。如果能减少搜索的节点数目，搜索效率会相应地提高。

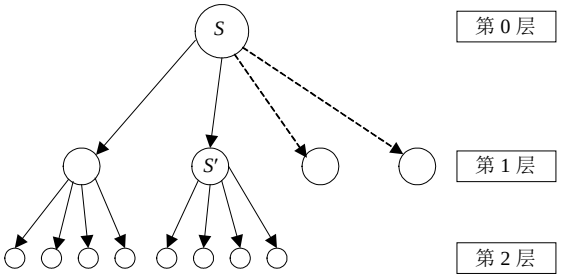

图1 增强型爬山法生成的搜索树

2.2 状态适应度函数

在启发式搜索过程中，候选节点的顺序将对搜索效率产生影响[8]。尽管FF利用多种技术提高了搜索效率，但FF在选择子节点进行评估时，并未充分考虑状态之间的优先关系，导致在搜索的过程中评估了过多的不利节点。针对增强型爬山法搜索的特点，如果能在进行搜索之前，按照某种性质对所有加入到队列中的状态给出一个大致的优劣判断(排序)，让最可能的状态节点放在队列最前端，使EHC搜索时能最快找到满足条件的节点，就能减少搜索的次数，减少无关状态的评估，提高搜索的效率。

实际上，FF的放宽规划图中隐含了大量有用的领域知识，为快速评估和比较后继状态提供了理论上的依据和可能性。从放宽规划图可以发现，动作增加表add(a)与子目标第一层G_1交集的元素越多，则执行该动作所生成的搜索状态越有可能接近目标状态，就可以将这类状态排列在扩展优先队列的前面。据此，本文提出了状态适应度函数的概念，它的形式化定义如下：

$$E(s_0,a,s)=\begin{cases}|\{p\mid p\in \mathrm{add}(a)\wedge p\in G_1\}| & s=\mathrm{exec}(s_0,a)\\ 0 \quad \text{否则}\end{cases}$$

式中，s_0是当前正在扩展的状态节点；a和s分别是状态s_0的有利动作集合中的动作及其对应的后继状态节点；G_1是放宽规划任务(P, A^+, s_0, G)的放宽规划图的子目标第1层。

状态s的适应度函数$E(s_0, a, s)$的值表明了该状态后续可扩展的适应程度，该值越大，表明其越可能接近目标状态，后续应考虑将其优先扩展。

2.3 基于有序爬山法的前向启发式搜索规划

本文以状态适应度函数为基础，结合增强型爬山法搜索后继状态的贪婪选择机制，提出了一种改进的局部搜索算法——有序爬山法，其核心思想是根据状态适应度函数对所有待扩展的后继状态进行排序，并加入到扩展优先队列。在启发式估值阶段，适应度高的状态将被优先计算评估，有利于更快地发现较优状态，从而减少调用启发式估值程序的次数。基于有序爬山法的前向启发式搜索的总体框架如图2所示。

```
Let π : = < >;   //solution plan
h(S): the heuristic value of state S;
HP(S): the set of helpful actions for S

Function OrderedHillClimbing (F, A, I, G)
1   S : = I;
2   while h(S) ≠ 0 and h(S) ≠ ∞ do
3       B : = OrderedSearchBetterState(S);
4       if   B == S   then
5             return failure;   //local minimal
6       else
7             S : = B ; // a better state is found
8    if h(S) == 0   then   return   π;
9    else        return   failure;
10  end

Function OrderedSearchBetterState (S)
1   Q : = SortNeighborhood (S);
2   while Q is not empty do
3       e : = remove first element of Q;
4       if   h(e) < h(S)     then
5             π : = π + extract_subplan (e, S);
6             return e; // return a better state
7       else if   h(e) ≠ ∞
8            N : = SortNeighborhood (e);
9            add N to the tail of Q;
10  return S; //return local minimal
11  end

Function SortNeighborhood(S)
1   N : = an empty queen;
2   foreach a ∈ HP(S), s = exec(S, a)
3        compute E (S, a, s);
4        Insert s into N with descending order according to E(S, a, s);
5   return N;
6   end
```

图2 基于有序爬山法的前向启发式搜索规划框架

函数OrderedHillClimbing从初始状态开始局部搜索(第3行)，如果遇到局部极小值，则返回局部搜索失败(第4～5行)，系统将调用贪婪最佳优先算法进行全局系统搜索；否则，找到一个更好的状态，系统循环进入到下一次局部搜索迭代过程(第6～7行)。

函数OrderedSearchBetterState是寻找较优状态的过程。该过程将维持一个先进先出的优先队列Q保存待扩展的状态，如果队列中的第1个状态是较优状态，则返回该状态(第4～6行)；否则，将第1个后续子状态全部加入到队列尾部(第7～9行)，一直到队列为空，表示没有找到较优状态，则返回局部极小值(第10行)。

函数SortNeighborhood将扩展当前状态的所有子状态，并利用状态适应度函数对这些子状态进行排序，加入到子状态优先队列，并返回该队列(第2～5行)。

3　实验和分析

根据本文所述思想，在FFv2.3规划系统的基础上实现了基于有序爬山法的规划系统(ordered hill climbing based planner, OHCP)。本文采用了国际规划大赛中，2个标准测试领域的STRIPS语言版本作为测试数据，这2个测试领域分别是Depots和FreeCell。这些领域的测试问题数目分别为22个、20个。测试平台为Fedora Release-8(Linux kernel 2.6.23)，编译器为g++(GCC)4.1.2，内存为1 GB，CPU为Intel Core Duo T2450 2.0 GHz。对每个领域问题，设定规划超时限制为180 s。

表1和表2分别列出了OHCP和FFv2.3的实验数据对比。表中，N为Expanded Nodes，T为Running Time，L为Plan Length，TO为Time Out。从表1可以看出，在Depots领域的大部分问题中，通过搜索节点数和搜索时间可得，OHCP的规划性能都比FF好(搜索节点数是最准确客观的衡量标准)；而且从规划解的长度可得，OHCP的规划质量在很多问题中比FF好或者与FF相当。FreeCell领域的实验数据对比如表2所示。OHCP在多数问题中比FF的规划性能更好或者相当。在较简单的问题中(即编号较小的问题)，OHCP的整体规划性能比FF更好；但在较复杂的问题中(即编号较大的问题)，OHCP的性能部分比FF较差。OHCP利用有序爬山法求解问题14～20归于失败，因此，OHCP转而采用系统化搜索算法GBFS重新求解这些问题，从而导致搜索节点数增加，搜索开销增大。FF在问题9、13、15、18～20也出现了类似的局部搜索失败的情形，从而同样会导致搜索开销的增大。出现这种情况的原因在于Freecell领域存在不可识别的死节点[7]，而放宽规划图启发式只是对状态的近似估计，有时它会将死节点(dead end)作为可扩展状态进行搜索，从而不可避免地导致局部搜索陷入困境(即遇到局部极小值)。

表1　Depots领域的实验对比

领域规划器问题	Depots					
	OHCP			FF		
	N	*T*	*L*	*N*	*T*	*L*
1	19	0.01	10	20	0.00	10
2	23	0.00	15	33	0.00	15
3	283	0.04	38	318	0.03	37
4	—	TO	—	—	TO	—
5	36 719	9.47	72	220 433	67.2	72
6	—	TO	—	—	TO	—
7	86	0.01	25	148	0.01	27
8	332	0.09	37	856	0.15	37
9	2 376	1.34	73	2 356	1.36	75
10	137	0.03	29	1 519	0.22	34
11	930	0.49	60	574	0.29	63
12	4 013	4.14	93	5 008	5.14	94
13	45	0.01	26	79	0.02	26
14	392	0.18	37	427	0.2	37
15	18 611	19.02	95	22 421	22.67	85
16	189	0.07	32	108	0.04	28
17	108	0.06	24	1 600	1.15	38
18	412	0.58	62	533	0.72	60
19	385	0.23	45	430	0.25	47
20	10 335	16.44	94	6 927	11.22	98
21	82	0.11	32	104	0.19	32
22	4 817	39.44	108	4 524	37.29	102

表2　Freecell领域的实验对比

领域规划器问题	Freecell					
	OHCP			FF		
	N	*T*	*L*	*N*	*T*	*L*
1	10	0.00	9	33	0.01	9
2	18	0.00	15	24	0.01	17
3	19	0.01	18	40	0.02	23
4	52	0.03	29	88	0.04	33
5	47	0.03	33	80	0.05	35
6	56	0.05	36	121	0.12	39
7	372	0.42	51	220	0.23	57
8	141	0.18	50	146	0.21	59
9	61	0.08	45	434	0.56	49
10	100	0.22	55	237	0.5	61
11	647	1.43	81	613	1.37	82
12	56	0.09	49	97	0.18	52
13	853	2.85	70	1 983	6.1	83
14	554	1.45	63	473	1.39	69
15	11 288	46.09	76	10 890	43.42	76
16	1 927	6.08	71	271	0.95	71
17	1 713	6.44	79	352	1.96	81
18	2 250	11.64	108	2 281	11.76	110
19	—	TO	—	—	TO	—
20	—	TO	—	—	TO	—

由于本文的方法对状态进行了预排序，存在将死节点优先进行扩展的情况，导致局部搜索失败。进一步的工作将研究如何使OHCP更好地处理局部搜索失败的情形。从规划解质量的角度，OHCP显然优于FF。通过实验观察发现，在大部分FreeCell领域的规划问题中，OHCP的最大搜索深度一般都不超过

FF，表明在有序爬山法的指引下，每次启发式搜索过程一般都能在更浅的搜索树层次找到较好的状态，既有利于更快地搜索到目标状态；更重要的是它能缩短搜索路径，从而减少了规划解的长度，提高了规划解质量。

从理论上分析，假定搜索树中每个节点的子状态平均为K个，需要搜索到最大为L层时成功，则第L层共有节点数K^L个。理想情况下，假设通过有序爬山法排序后，在第L层第一个元素就能搜索成功。而在增强型爬山法引导下，假设每个节点具有同等概率成为较好的节点，则平均需搜索第L层一半的节点才能搜索成功。理想情况下，通过有序爬山法可减少搜索的节点数目为$K^L/2$，则搜索效率的提升比例为$K^L/2(K^0+K^1+\cdots+K^L)$，即减少搜索的节点数与增强型爬山法总搜索节点数的比值。

由此可知，当搜索层数较小，即L值较小时，比值越大，有序爬山法的效果更明显；而当搜索层数越深，则L值越大，相对来说有序爬山法的作用趋于弱化。

4 结 束 语

启发式搜索技术已被很多高性能规划器所采用[3-6]。提高启发式估值的信息含量(informedness)、加快搜索算法的速度、减少存储空间消耗等是启发式搜索算法的主要研究方向[12]。FF在这些方面做出了开创性的成果，基于放宽规划图思想的前向启发式搜索方法成为当前的研究热点，而FF中使用的增强型爬山法也被其他规划器广泛引用，显示出强大的搜索能力。本文充分利用了领域知识，通过动作和放宽规划图之间存在的内在联系，计算出状态适应度用以快速评估和比较扩展状态的优劣；以此为基础，对搜索过程中的状态进行排序，从而能减少搜索的节点数目，提高搜索效率。爬山法作为一种常见的启发式搜索算法，被应用于很多规划器中，未来的工作可以扩展到采用贪婪搜索算法和基于放宽规划图的其他类型规划器中[13]，并扩展其非STRIPS规划语言类型，如ADL[14]。另一方面，状态适应度函数仍然可以继续精化，使其包含更丰富和有用的信息量，但这可能耗费更多的计算时间。如何在时间效率和质量两个维度取得平衡，这是一个值得研究的课题。

5 致谢

本文得到了中山市科技计划项目(20123A314, 20123A327, 411S16)的资助，在此表示感谢。

参 考 文 献

[1] GHALLAB M, NAU D S, TRAVERSO P. Automated planning: Theory and practice[M]. San Francisco, USA: Morgan Kaufmann, 2004.

[2] LONG D, FOX M. The 3rd International planning competition: Results and analysis[J]. Journal of Artificial. Intelligence Research, 2001(20):1-59.

[3] BONET B, GEFFNER H. Planning as heuristic search[J]. Artificial Intelligence, 2001, 129(1-2): 5-33.

[4] HOFFMANN J, NEBEL B. The FF planning system: Fast plan generation through heuristic search[J]. Journal of Artificial Intelligence Research, 2001(14): 253-302.

[5] HELMERT M. The fast downward planning system[J]. Journal of Artificial Intelligence Research, 2006(26): 191-246.

[6] RICHTER S, WESTPHAL M. The LAMA planner: guiding cost-based anytime planning with landmarks[J]. Journal of Artificial Intelligence Research, 2010(39): 127-177.

[7] HOFFMANN J. Where "ignoring delete lists" works: Local search topology in planning benchmarks[J]. Journal of Artificial Intelligence Research, 2005(24): 685-758.

[8] REINEDELD A, MARSLAND T A. Enhanced iterative-deepening search[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1994, 16(7): 701-710.

[9] FIKES R, NILSSON N. STRIPS: A new approach to the application of theorem proving to problem solving[J]. Artificial Intelligence, 1971, 2(3-4): 189-208.

[10] 梁瑞仕. 基于子目标排序和启发式搜索的若干规划技术研究[D]. 广州: 中山大学, 2010.
LIANG Rui-shi. Research on AI planning techniques based on subgoal ordering and heuristic search[D]. Guangzhou: Sun Yat-Sen University, 2010.

[11] BLUM A, FURST M. Fast planning through planning graph analysis[J]. Artificial Intelligence, 1997, 90(1-2): 281-300.

[12] HELMERT M, DOMSHLAK C. Landmarks, critical paths and abstractions: What's the difference anyway [C]// Proceedings of the 19th International Conference on Automated Planning and Scheduling(ICAPS'09). Thessaloniki, Greece: AAAI, 2009.

[13] GEREVINI A, SAETTI G A, SERINA I. Planning through stochastic local search and temporal action graphs in LPG[J]. Journal of Artificial Intelligence Research, 2003(20): 239-290.

[14] KOEHLER J, NEBEL B, HOFFMANN J, et al. Extending planning graphs to an ADL subset[C]//Recent Advances in AI Planning, 4th European Conference on Planning (ECP'97). Toulouse, France: Springer, 1997.

编 辑 黄 莘