

局部满意的启发式搜索算法

谢琳

(江苏教育学院 苏州学前教育分院, 江苏 苏州 215008)

摘要: 在经典规划中, 目标是找到一系列连续的行为, 改变初始状态 Z 到一些满意的目标状态 G . 局部满意规划 (PSP) 问题是规划问题中的核心问题之一. 在 PSP 中, 文献[1-2] 给出的每个目标有一个功能值 $u_g \geq 0$, 代表每个目标对于用户的价值; 每个行为 $a \in A$, 有一个关联执行代价 $C_a \geq 0$, 代表它执行每个行为的代价. P 为所有有效规划集, $G_p \in G$ 为目标集, 目标是寻找一个规划 p 在功能 u_g 和执行代价之间寻找最大差, 即 $\arg \max_{p \in P} \sum_{g \in G_p} u_g - \sum_{c \in p} C_a$. 针对局部满意问题, 提出了一种新的启发式搜索算法. 该算法经过验证, 取得了明显的效果.

关键词: 启发式搜索算法; 局部满意规划; 宽松规则; 向前状态启发式搜索算法

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2011)10-0194-07

Heuristic Search for Partial Satisfaction Algorithm

XIE Lin

(College of Preschool of Suzhou, Education University of Jiansu, Suzhou 215008, China)

Abstract: In classical planning, the aim is to find a sequence of actions that transforms a given initial state Z to some state satisfying goals G . Partial satisfaction planning is one of key point in planning problem. In partial satisfaction planning^[1-2], each goal has a utility value $u_g \geq 0$, representing how much each goal is worth to a user; each action $a \in A$ has an associated execution cost $C_a \geq 0$, representing how costly it is to execute each action. Let P be the set of all valid plans and let $G_p \in G$ be the set of goals achieved by a plan. The objective is to find a plan P that maximizes the difference between total achieved utility u and total cost of all actions: $\arg \max_{p \in P} \sum_{g \in G_p} u_g - \sum_{c \in p} C_a$. This article gives a new heuristic search algorithm for partial satisfaction planning. It confirms their effective through the examples.

Key words: heuristic search algorithm; partial satisfaction plan; loose rule; forward condition heuristic search algorithm

1 引言

例: 有一辆卡车装着 3 个包裹. 开始在 A 处, 软目标是运送 1 个包裹到其他 4 个位置, 每地一个; B 、 C 、 D 、 E 各个目标的功能值见图 1. 例如: $g_1 = \text{HavePacket}(B)$ 和 $u_{g_1} = 50$. 为了简化问题, 我们假设有一个单独的行为 $\text{Move}(X, Y)$, 移动卡车从地点 X 到地点 Y , 到达目标 Y , 卡车卸下一个包裹在 Y . 如果卡车上有一个包裹, 它将在 Y 卸下. 如果卡车上没有包裹, 但在 X 处有一个, 卡车将捡起它, 并把它在 Y 处放下, 执行 $\text{Move}(X, Y)$. 在两个不同地点移动

的代价, 如图 1 所示. 如果 $a_1 = \text{Move}(A, B)$, $c_{a_1} = 40$ (注意 $c_{\text{Move}(X, Y)} = c_{\text{Move}(Y, X)}$), 目标函数是寻找一个规划, 给出在建立总的目标功能和移动代价之间最佳的交换. 在此例中, 最佳的规划可表示为, $P = \{\text{Move}(A, C), \text{Move}(C, D), \text{Move}(D, E)\}$, $g_1 = \text{HavePacket}(B)$, 当建立最后 3 个目标 g_2, g_3, g_4 时, 可忽略第 1 个目标, 我们称这种功能和代价交换值为“网络增益”, 即 PSP 的满意规划问题. 关于这方面的工作, 文献[3-6] 取得了长足进展, 但也还有许多问题有待进一步探索, 如局部满意问题. 因此本文主要围绕此问题进行研究.

收稿日期: 2011-04-28; 修回日期: 2011-06-27

基金项目: 国家自然科学基金项目(60775045, 61033013)

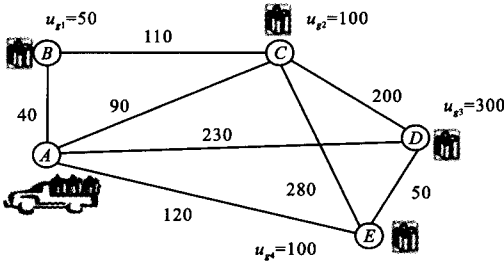


图 1 一个物流的例子

2 宽松规划

在文献[7-9]的基础上,本文提出宽松规划来近似最短长度规划(例如:最少量的动作数)来建立目标.

宽松规划问题是忽略所有负效果的动作. 其解决方法是 P^+ 对于宽松规划问题 (F, A^+, I, G) , 从 A 中建立 A^+ , 从所有动作 $a \in A$ 中移去负效果 $Delete(a)$, 动作和事实数在多项式时间里能找到. P^+ 被称为一个“宽松规划”. 通用的方法是提取 P^+ , 可分 2 步解决.

- (1) 建立宽松规划图 PG^+ , 使用 A^+ , 从 I 向前.
- (2) 从 PG^+ 中提取 P^+ , 从 G 向后.

在第一步, 动作和事实水平是交叉的, 从 $I = F_0$ 开始作为第一事实水平. 第 i 动作和事实水平 A_i 和 F_i 的建立反复使用以下规则:

$$A_i = \bigcup \{a: Pre(a) \subseteq F_{i-1}\} \tag{1}$$
$$F_i = F_{i-1} \cup (\bigcup_{a \in A_i} Add(a)) \tag{2}$$

这些等式, 可以用程序进行描述, 当 $G \subseteq F_i$ 时, 扩展停止. 开始子目标集 $SG = G$, 在最后的 i 层扩展停止, 宽松规划 $P^+ = \varphi$. 对于每一个 $g_i \in SG$ (例如目标 g 在 i 层), 我们选择一个支持动作 a_i (例如动作 a 在 i 层). 如 $g \in Effect(a)$, 并且更新 $P^+ \leftarrow P^+ \cup \{a_i\}$ $SG \leftarrow SG \cup \{P_{i-1} : p \in Pre(a)\}$. 注意, 子目标 g_i 也能被一个 noop 动作在 i 层支持, 如果 g 在图中的 $i-1$ 层出现. 在此例中, 只是在 SG 用 g_{i-1} 取代. 重复步骤, 直到 $SG \subseteq F_0$.

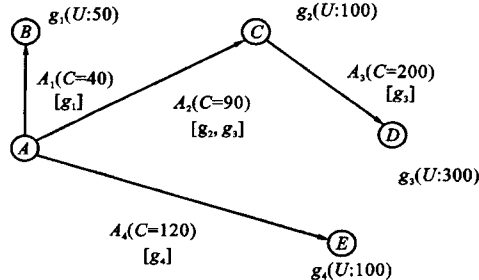


图 2 宽松规划图

如图 2 所示, 第 1 个事实水平代表初始状态, 仅仅 $at(A)$ 是真. 第一个动作层包含在初始状态合适的动作, 有 4 个 Move 动作, 从 A 到其他 4 个地点. 第 2 个事实层包含第 1 层的动作效果. 第 2 动作层包含的动作是第 2 事实层前提满足的, 并且这些包含问题中的所有动作. 例如图中提取的宽松规划是 $P^+ = \{Move(A, B) Move(A, C), Move(A, E), (C, D)\}$ 从这里可以看到这宽松规划在相互冲突的关系 $Move(A, C), Move(A, B), Move(A, E)$ 之间是不一致的. 无论如何, 它可以对一个真正的一致规划作出估价. 因此, 如果目标函数是寻找最低代价规划, 那么, 从图 2 中提取的初始状态的预计代价

$$h(I) = C_{Move(A, B)} + C_{Move(A, C)} + C_{Move(A, E)}$$
$$+ C_{Move(C, D)}$$

如果目标函数是寻找最短长度规划, 则 $|P^+| = 4$ 可以作为启发式的值.

3 向前状态空间启发式规划

3.1 PSP 的向前宽松优先搜索算法

目前的向前状态空间搜索算法所描述的问题, 文献[3-8]在此方面做了不少工作. 其向前状态空间搜索算法描述为算法 1.

算法 1: 向前状态空间搜索算法

```
1  Input: A planning problem:
2  Output: A valid plan:
3  begin
4      OPEN ← {I}
5      while OPEN ≠ ∅ do
6          s ← arg min f(x);
              X ← OPEN
7          OPEN ← OPEN \ {s};
8          If s = G then
9              return plan leading to s and stop
              search;
10         else
11             foreach a applicable in s do
12                 OPEN ← OPEN ∪ {Apply(a, s)}
13             end
14         end
15     end
16 end
```

本文将引入启发式技术进行研究, 将向前状态空间搜索问题描述如下:

- (1) 每个搜索节点 n 代表一个完整规划状态 s .

(2) 如果申请一个动作 a 到一个状态 s 导致另一个状态 s' , 那么动作 a 代表边 $e = s \xrightarrow{a} s'$, 从 s 到 s' 边代价 $c_e = c_a$.

(3) 开始节点代表初始状态 I .

(4) 一个目标节点是任何状态 s_G 满足所有目标 $g \in G$.

在本文的例子中, 初始状态 $I = \{at(A)\}$, 有 4 个待处理的动作 $a_1 = \text{Move}(A, B)$, $a_2 = \text{Move}(A, C)$, $a_3 = \text{Move}(A, D)$, $a_4 = \text{Move}(A, E)$, 导致 4 个状态 $s_1 = \{at(B), g_1\}$, $s_2 = \{at(C), g_2\}$, $s_3 = \{at(D), g_3\}$, $s_4 = \{at(E), g_4\}$. 边的代价在规划状态过渡图中将代表动作代价, 最短路径在图中代表最低代价规划, 与经典规划问题相比, PSP 问题差别表现在:

(1) 不是所有目标需在最终规划中完成.

(2) 目标不是统一的, 具有不同的功能值. 规划的质量不是测量总的动作代价, 而是目标们建立的积累功能值和动作使用积累的代价之间的区别. 因此, 目标函数在最小的总的动作代价与最大的网络增益之间切换.

下面给出本文的改进方案:

g 值: A^* 使用 $f(s) = g(s) + h(s)$ 的值排列产生的状态扩展, g 代表了(已知)路径的值引导开始状态 I 到 s , h 估计(未知)未知路径从 s 到一个目标节点, 这将最优化一个给定的目标函数. 在 PSP 问题中, g 代表从路径 I 到 s 额外的利益获得. 对于一个给定的状态 s , 使 $G_s \in G$ 是在 s 中建立的目标集, 那么

$$g(s) = (U(s) - U(I)) - C(P_{I \rightarrow s}) \quad (3)$$

$U(s) = \sum_{g \in G_s} u_g$ 和 $U(I) = \sum_{g \in G_I} u_g$ 是在满意的 s 和 I 时的总的目标功能. $C(P_{I \rightarrow s}) = \sum_{a \in P_{I \rightarrow s}} C_a$ 是在 $P_{I \rightarrow s}$ 时, 总的动作的代价. 例如, $U(s_2) = u_{g_2} = 100$, $C(P_{I \rightarrow s_2}) = C_{a_2} = 90$, 并且 $g(s_2) = 100 - 90 = 10$.

另外, $g(s)$ 如式(3)定义代表额外的增益, 当 $P_{I \rightarrow s}$ 代表 I 到达 s . 为便利讨论, 我们将介绍一个新的标记来代表一个规划 p 从一个状态 s 到另一个状态 s' :

$$B(P | s) = (U(s') - U(s)) - \sum_{a \in p} C_a \quad (4)$$

这样, 我们有 $g(s) = B(P_{I \rightarrow s} | I)$

h 值: 在图搜索中, 启发式的值 $h(s)$ 估计从 s 到“最佳”目标节点的路径. 在 PSP 中, “最佳”的目标节点是节点 s_g , 从 s 到 s_g 将给出最大的额外增益. 通常, h 估计真正的最优 h^* 的值越接近, 总的搜索效

果就越好. 因此, 我们将首先证明 h^* .

最佳获益规划:

对于一个给出的状态 s , 最佳增益规划 P_s^B 在 s 执行, 并且没有其他规划 p 如 $B(P | s) > BCP_s^B | s$.

注意: 空规划 $P_\emptyset$ 表示在所有状态没有动作执行, 并且 $B(P_\emptyset | s) = 0$. 因此, 任何状态 s 有 $B(P_s^B | s) \geq 0$. 一个被给出的状态 s 最优额外建立的增益由如下计算:

$$h^*(s) = B(P_s^B | s) \quad (5)$$

算法 2: 改进的向前状态空间搜索算法

```

1  Input: A PSP problem:  $\langle F, I, G, A \rangle$ ;
2  Output: A valid plan  $P_s$ 
3  begin
4       $g(I) \leftarrow g \in I \cap \Sigma u_g$ 
5       $f(I) \leftarrow g(I) + h(I)$ 
6       $B_s \leftarrow g(I)$ 
7       $P_s \leftarrow \phi$ 
8      OPEN  $\leftarrow \{I\}$ ;
9      while OPEN  $\neq \phi$  and not interrupted do
10          $s \leftarrow \arg \max f(x)$ 
11         OPEN  $\leftarrow \text{OPEN} \setminus \{s\}$ ;
12         if  $h(s) = 0$  then
13             stop search;
14         else
15             foreach  $s' \in \text{Successor}(s)$  do
16                 If  $g(s') > B_s$  then
17                      $P_s \leftarrow \text{plan leading from } I \text{ to } s'$ 
18                      $B_s \leftarrow g(s')$ 
19                     OPEN  $\leftarrow \text{OPEN} \cup \{s' | f(s') \leq B_s\}$ 
20                 end
21                 If  $f(s') > B_s$  Then
22                     OPEN  $\leftarrow \text{OPEN} \cup \{s'\}$ 
23                 end
24             end
25         end
26     end
27     Return  $P_s$ ;
28 end

```

在本文的例子中, 从状态 s_2 , 最有益处规划 $P_s^B = \{\text{Move}(C, D), \text{Move}(D, E)\}$

$$h^*(s_2) = B(P_s^B | s) = U(\{g_3, g_2, g_4\}) - U(\{g_2\}) - (C_{\text{Move}(C, D)} + C_{\text{Move}(D, E)}) = ((300 + 100 + 100) - 100) - (200 + 50) = 400 - 250 = 150$$

直接计算 h^* 是不现实的, 我们需要搜索 P_s^B 在可能的规划空间, 这和解决 PSP 问题一样困难. 一个好的近似值 h^* 需要有其他技术支持, 因此 本文主要讨论启发式近似 P_s^B .

算法 2 描述了任何时间变种的 A^* 算法. 在 A^*

中,使用 $f = g + s$ 函数排列节点的扩展.

3.2 PSP 的宽松规划启发式搜索算法

给一个搜索节点相当于一个状态 s , 启发式需要估计状态 s 的网络 z 增益走向, 在搜索中基于从宽松规划抽取的思想. 我们开始注意为了确保任意时间搜索找到最优解, 估计网络的 benefit-to-go 必须是 s 的网络 benefit-to-go 为真的上界. 为了确保这, 我们必须找到“最优” P^+ 规划. 为解决从状态 s 的原始问题的宽松版本, 这将是 NP 难题, 甚至对经典规划. 寻找最优宽松规划是 NP 难题. 因此, 我们将牺牲可容许的启发法, 把焦点放在寻找多项式时间里“good”宽松规划.

3.2.1 聚集宽松规划的建立

在聚集方法中, 我们通过一个 NULL 规划计算贪心宽松规划. 在一个时间里扩展它以支持另一个目标. 查到宽松规划的网络增益开始减少. 关键问题是目标被考虑的次序. 通过单独的网络增益顺序排序来选择第一个目标是合理的. 因此, 对每个目标 $g = G$, 我们计算“贪心的”宽松规划 P_g^+ 来支撑 g (使用向后清理方法). 此目标 g 被估计的网络增益由 $u_g - C(P^B)$ 给出.

3.2.2 PSP 的剪枝宽松规划通用法的描述

让我们首先形式化问题以寻找一个最优的规划的子集.

定义(合适的子规划): 给出一个规划 P 作为一系列动作(严格说类型 s_i 的“步子”; a_j 一个基本动作能在规划的多个时间段) $a_1 \cdots a_n$, P' 被考虑为一个合适的 P 的子规划, 如果 P' 是 P 的子集(随后到达的通过移去 P 中的一些元素), 并且 P' 是一个有效的规划(能被执行)

3.2.3 最有益的合适的子规划

一个合适的子规划 P_0 是 P 中最有益的子规划, 如果在 P 中没有其他合适的子规划 P' , 它的增益高于 P_0 .

命题 1: 寻找最有益的合适的子规划是一个 NP 难题.

证明: 最小化“经典”规划问题(例如, 移去动作仍然保持它的正确)能被分解寻找最佳网络增益的子规划问题. 过去是 NP 难题^[10].

当我们不打算寻找最有增益的合适的子规划, 理解它怎样做仍然是有益的. 下面, 我们提供了一个从规划中识别和移去动作的标准, 而不减少它的网络益处.

3.2.4 单独——支持动作集

G_P 由 P 建立的目标, $A_G \subseteq P$ 是一个单独支持动作集 $G' \subseteq G_P$, 如果:

- (1) 余下的规划 $P_r = P \setminus A_G$ 是一个有效的规划
- (2) P_r 在 $G_P \setminus G'$ 建立所有目标
- (3) P_r 在 G' 不建立所有目标

(4) 没有动作子集 $A' \subseteq P_r$ 从 P_r 中移去, 如果 $P_r \setminus A'$ 在 $G_P \setminus G'$ 中建立所有目标是一个有效的规划.

注意: 在 P 中可能存在额外的动作, 没有为任何目标的建立作贡献, 而那些动作可作为空的目标集 $G' = \emptyset$ 的单独支撑来考虑. 上面最后的条件确保那些动作包含在任何支撑动作集 A_G 因此剩余的规划 P_r 是尽可能有益的. 我们定义没有益处的目标集和支撑动作如下:

3.2.5 无益处目标集

规划 P 建立目标集 G_P , 目标集 $G' \subseteq G_P$ 是一个无益处的目标子集, 如果具有一个单独支持动作集 A_G 并且总的动作代价在 A_G 中高于在 G' 中的全部目标功能.

这很明显, 移去无益处目标集 G' 和它的单独支持动作集 A_G 能仅仅提高规划的总的益处. 因此, 如果 $P_r = P \setminus A_G$ 那么

$$\sum_{g \in (G_P \setminus G')} u_g - \sum_{a \in P_r} c_a \geq \sum_{g \in (G_P)} u_g - \sum_{a \in P} c_a \quad (6)$$

规划 P 的建立目标 G_P , 最佳方法剪枝是移去最没有益处的目标集 G' 这能导致剩余的规划有最高增益. 因此, 我们寻找和移去 $G' \subseteq G_P$ 和它的单独支持动作集 A_G 如:

$$G' = \operatorname{argmin}_{G' \subseteq G_P} \sum_{g \in G'} u_g - \sum_{a \in A_G} c_a \quad (7)$$

命题 2: 给出一个规划 P , 如果 G' 它的单独支撑动作集 A_G 是最无益的目标集作为式(7)指定的, 那么 $P' = P \setminus A_G$ 是子规划 P 中最有益的可能子规划.

证明:

设 P_0 为 P 的最优增益的子规划, B, B_0, B' 分别代表 P, P_0 和 P' 中的网络增益. 设 P_0 是最优的, 我们有 $B_0 \geq B'$ 设 G_0 是有规划 P_0 建立的, 我们定义 $A_0 = P \setminus P_0$ 和 $G'_0 = G \setminus G_0$. 我们想说明 A_0 是单独支撑集 G'_0 中的. 设 P_0 是一个有效的规划, 建立 G_0 中的所有目标, 并且在 G'_0 中一个也没有. 前三个条件对于 A_0 是 G'_0 的单独支撑是显然满足的. 设 P_0 是 P 的最有益处的子集, P_0 中没有任何子集能建立 G_0 中所有目标. 因此, 第四个和最后的条件也是满足的. 因此, A_0 是 G'_0 中的单独支撑集. 设 G'_0 是最没

有益处的目标集, 等式(7) 显示

$$B_1 = \sum_{g \in G'} u_g - \sum_{a \in A_{G'}} c_a \geq B_2 = \sum_{g \in G'} u_g - \sum_{a \in A_0} c_a$$

因此, $B' = B + B_1 \geq B_0 = B + B_2$, 与 $B_0 \geq B'$ 合并, 能得到以上分析.

我们有 $B' = B_0$, 因此, P' 是 P 中最有益处的子集.

执行规划中的一个增长算法来近似鉴定和移走宽松规划中最没有益处的目标集和它联合的单独一支撑动作集. 由于我们申请的宽松规划的程序代替真正的规划, 仅有一个条件需要改变, 剩余规划 $P' = P_G \setminus AG'$ 仍然是一个有效的宽松规划. (例如, 忽略删除列表时能执行.)

这扫描和过滤进程包含 2 个主要步骤:

(1) 鉴定: 使用宽松规划的因果关系—结构, 识别集合的顶层目标和每个动作支撑. 如算法 4 所示.

(2) 增长移去: 使用第一步计算的支撑目标集, 启发式的移走没有益处的动作和目标, 如算法 4 所示.

第一步使用的支撑目标列表 GS , 为每个动作 a 和事实 p 定义如下:

$$GS(a) = \bigcup_{p \in \text{Effect}(a)} GS(p) \quad (8)$$

$$GS(p) = \begin{cases} p \cup \left(\bigcup_{p \in \text{Prec}(a)} GS(a) \right) & \text{if } p \in G \\ \bigcup_{p \in \text{prec}(a)} GS(a) & \text{if } p \notin G \end{cases} \quad (9)$$

在算法 4 的执行中, 开始

$$\forall p \notin G: GS(a) = GS(p) = \emptyset$$

$$\text{并且 } \forall g \in G: GS(g) = \{g\}.$$

上面列出的 2 条规则申请重复执行, 从顶层目标开始, 直到 $GS(a)$ 或者 $GS(p)$ 集都没有改变. 因此, 对每个动作 a , 我们知道哪个目标是它支持的.

提取的宽松规则

$$P^+ = \left\{ \begin{array}{l} a_1: \text{Move}(A, B), a_2: \text{Move}(A, C) \\ a_3: \text{Move}(C, D), a_4: \text{Move}(A, E) \end{array} \right\} \text{ 和目}$$

标一样每个动作都支持. 从顶层目标回溯, 动作 a_1, a_3, a_4 仅支持 g_1, g_3, g_4 , 所以, 这些动作的目标支持列表将是 $GS(a_1) = \{g_1\}, GS(a_3) = \{g_3\}, GS(a_4) = \{g_4\}$ 在 c 点, 动作 a_3 的前提条件将依次建立目标 g_3 . 因此, 我们有 $GS(at(C)) = \{g_3\}$. 最后, 因为 a_2 支持 g_2 并且在地点 c , 我们有 $GS(a_2) = GS(g_2) \cup GS(at(C)) = \{g_3, g_2\}$.

在第 2 步, 使用每个动作的支撑目标集, 我们能识别动作子集 $SA(G')$, 仅对目标建立, 在给出的目标算法 3: PSP 的启发式搜索算法

```

1  Input: A relax plan  $P'$  achieving the goal set  $G_p$ ;
2  Output: The supported goal set for each action
    $a \in P'$  and fact:
3  begin
4       $\forall a \in P: GS(a) \leftarrow \emptyset$ 
5       $\forall g \in G_p: GS(g) \leftarrow \{g\}$ 
6       $\forall p \in G_p: GS(p) \leftarrow \emptyset$ 
7       $DONE \leftarrow \text{false}$ 
8      while not  $DONE$  do
9           $DONE \leftarrow \text{true}$ 
10         forall  $a \in P'$  do
11             if  $GS(a) \neq \bigcup_{p \in \text{Effect}(a)} GS(p)$  Then
12                  $GS(a) \leftarrow \bigcup_{p \in \text{Effect}(a)} GS(p)$ 
13                  $DONE \leftarrow \text{false}$ 
14             end
15         end  $P'$ 
16         forall Proposition  $p$  do
17             if  $GS(p) \neq GS(p) \cup \left( \bigcup_{p \in \text{Effect}(a)} GS(a) \right)$  then
18                  $GS(p) \leftarrow GS(p) \cup \left( \bigcup_{p \in \text{Effect}(a)} GS(a) \right)$ 
19                  $DONE \leftarrow \text{false}$ 
20             end
21         end
22     end
23 end

```

标子集 $G' \in G$:

$$SA(G') = \bigcup_{GS(a) \subseteq G'} a \quad (10)$$

如果在 $SA(G')$ 总的动作代价超过了在 G' 的目标功能的总和 (例如, $\sum_{a \in SA(G')} c(a) \geq \sum_{g \in G'} u(g)$), 那么我们能从宽松规划中移去 G' 和 $SA(G')$. 我们称那些子目标集无益处. 在我们的例子中, 动作 a_4 对于建立 g_4 仅有一个贡献. 由于 $c(a_4) \geq u(g_4)$ 和 a_4 不对其他目标作贡献, 我们能将 a_4 和 g_4 从考虑中移去. 在 P^+ 中剩下的 3 个动作是 a_1, a_2, a_3 , 并且目标 g_1, g_2, g_3 都显示有增益的. 通常, 如果我们考虑所有 $2^{|G|}$ 可能的目标子集的潜在的移动是昂贵的. 因此, 在算法 4, 我们仅考虑目标子集 G' 如 $|G'| \leq N$, N 作为预先定义的值. 如果没有无益处的目标集 G' 且 $|G'| \leq N$, 那么我们终止剪枝算法. 如果有增益的目标集, 则我们选择增益最小的一个 G_x , 从当前宽松规划 P^+ 中移去动作和单独支撑的 G_x . 然后再次开始剪枝进程. 在我们当前的执行中, 我们仅考虑 $N = 0, N = 1$ 和 $N = 2$. 不管怎样, 如我们之后讨论的, 仅考虑 $N = 0, N = 1$ 常常是足够的.

算法 4:PSP 的宽松规划启发式搜索算法

```
1 Input: A relaxed plan and parameter      N>0
2 Output: A relaxed plan  P' ⊆ P* with higher or equal
    benefit compared to P*
3 begin
4   OPEN ← false
5   while not DONE do
6     Use Aglorithm 2 compute all GS(a),GS(p) and
        using P*
7     forall G' ⊆ Gp and |G'| ≤ N do
8       SA(G') ← ∪GS(a) ⊆ G' a
9     end
10    Gs ← argmin ( ∑ u(g) - ∑ C(a) )
        G' ⊆ Gp}, |G'| ≤ N  g ∈ G'  a ∈ SA(G')
11    if ( ∑ u(g) - ∑ C(a) ) ≤ 0 then
        g ∈ Gs}  a ∈ SA(Gs})
12      P' ← (P* \ SA(Gs}))
13    else
14      DOEN ← false
15    end
16  end
17 end
```

在移去无益处的目标和(单独)支撑它们的动作,剩余宽松规划的代价和目标功能,它的建立通常计算 h 值. 因此,在我们的例子中,

$$h(I) = (u(g_3) + u(g_2) + u(g_1)) - (c(a_2) + c(a_3) + c(a_1)) = (100 + 300 + 50) - (90 + 200 + 40) = 120$$

当启发式是有效的,它不能保证是允许的,因为原始的宽松规划是启发式提取的并且可能不是最优的. 此外,我们仅考虑移去的目标子集 G' 如 $|G'| \leq 2$. 因此,我们在算法中使用启发式(在前面的算法 2) 讨论的. 规划返回最优是没有保证的. 这是因为:

- (1) 一个剪枝节点确实引导一个最优规划. (例如,扩展到达节点 s 且 $g(s) > B_B$).
- (2) 最后的 B_B 可能不代表一个最优规划.

为了处理第一点,我们在执行中调整,以便即使权重 $w = 1$ 被使用在等式 $f = g + w \cdot h$ 来排序队列中的节点. 另一个值 $w = 2$ 被使用剪枝节点,且 $f = g + w \cdot h$. 对于第二点,我们继续搜索一个更好的规划,在节点 $h(s) = 0$ 之后被找到,直到一些资源限制到达. (例如,我们已到达一些数量的搜索节点或者被规划时间所限制.).

3. 2. 6 在前进搜索中使用宽松规划

为了使用最后的剪枝宽松规划来实现启发式一计算的目的,我们也执行了一个原始的前进技术,使用我们给出的宽松规划,在尽可能多的真实的规划问题中模拟执行(例如,规划者尝试每个动作,通过宽松规划的因果关系结构,按次序定义,重复这个进程,直到没有动作被执行). 对于规划者 YAHSP^[11] 的结果,这个技术是有创造力的. 它使用一个类似的但更加老练的前进战略. 我们发现这种方法有助于在更少的时间里找到更好质量的规划,但也使搜索到达终止搜索节点更快.

4 实验分析

本文主要选择两类实例,即卡车域和路径域问题,下面分别描述.

4. 1 卡车域

卡车域由运送包裹到不同地点的卡车组成. 这是“logistics”类型域,使用的约束是在装载到其他存储区之前,卡车的某些存储区面积必须是空的. 这个域的“简单参数”版本,包裹必须被运送到或者在某个时间之前,以避免引起一个参数违规的惩罚.

图 3 显示卡车域中规划问题的性能比较结果. Yochan^{ps}在这个域比其他规划者表现要好. 它有些地方比 MIPS-XXL^[12] 和 MIPS-BDD^[12] 两者都好. SGPLAN^[13]能解决更多问题,常常是更好或相等质量. 特别地,在问题 7 到 9, Yochan^{ps}很难找到好的质量的规划者. 在产生的问题之间检验不同提供一些行为的洞察. 在这个域的前 10 个问题中,参数的个数(例如,软目标)的增长作为问题大小增长的部分. 这些都包括在存在的量化来处理包裹运送的最后期限. 一个包裹必须被运送在规划的一个特别的编码时间之前(在一个单元里,当驾驶或运送包裹时间增长). 例如,包裹 1 可能需要被运送在一个时间步 t_3 之前. 因为这个标准的定义使用了一个谓词,这促使基础的数字,软的分离目标集增长. 这轮流促使更多目标在每个时间步中被考虑. 在这些环境中,这规划图的代价的传播及目标选择进程将花费更多时间. 作为比较,第二个问题集(问题 11 到 20),在目标参数上包含完全的包裹运送时间(例如,包裹 1 必须在确定的时间 t_5 被运送). 因此避免了分离参数的需要. 规划者解决了这些更艰难的问题中的 4 个实例.

4. 2 路径域

这个域它来源于分子生物学. 它的模型通过动作的化学反应,并且包含其他的动作选择最初的培

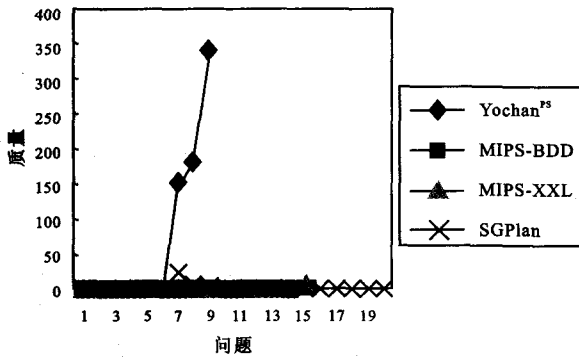


图3 Logistics域规划问题的求解性能对比

培养基. 这个域的目标在“简单参数”部分给物质一个参数, 其必须通过一个路径制造.

图4显示 Yochan^{PS}在这个域是趋于拙劣的. 尽管这很大程度上由于规划者在基础进程期间运行超出内存. 例如, 在问题5宣布的物体数引起我们基础程序在超过106个动作中企图制作较好. 在大多数它解决的问题上, Yochan^{PS}与其他规划者相比提供了相等的质量. 图4显示 Yochan^{PS}和 Yochan^{cost}两者创建的规划质量相等. 注意: 在比赛中, Yochan^{PS}和 Yochan^{cost}修补了一个小的 bug. 在这个域, 在新的运行时期内不能找到问题4的解答.

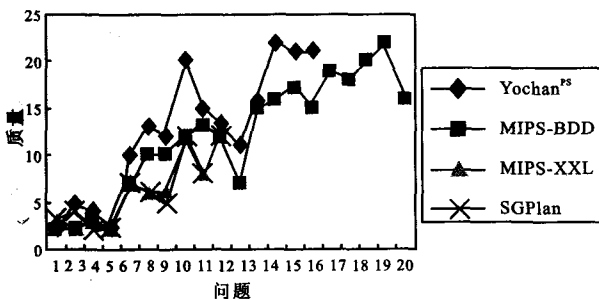


图4 路径域规划问题的求解性能对比

5 结束语

本文给出了一个启发式搜索算法来解决局部满意规划问题. 并将相关算法应用于具体实例域, 取得了满意的成绩. 不过也还有很多问题需进一步研究. 如能否将该成果扩展到全局问题等.

参考文献:

- [1] Smith D E, Choosing objectives in over-subscription planning[C]// Proceedings of ICAPS-04, 2004: 393-401.
- [2] Van den Briel M, Nigenda R S, Do M B, et al. Effective approaches for partial satisfaction (over-subscription) planning [C]// Proceedings of the 19th national conference on Artificial intelligence. [s. l.]: AAAI, 2004: 562

-569.

- [3] Baier J, Bacchus F, McIlraith S, A heuristic search approach to planning with temporally extended preferences [J]. Journal Artificial Intelligence, 2007, 173(5): 1808-1815.
- [4] Baier J, Hussell J, Bacchus F, et al. Planning with temporally extended preferences by heuristic search [C]// Proceedings of the ICAPS Booklet on the Fifth International Planning Competition UK: IEEE, 2006: 7-10.
- [5] Benton J, Do M B, Kambhampati S. Over-subscription planning with numeric goals [C]// Proceedings of the 19th international joint conference on Artificial intelligence San Francisco CA, USA: ACM, 2005: 1207-1213.
- [6] Benton J, Kambhampati S, Do M. YochanPS: PDDL3 simple preferences as partial satisfaction planning, [C]// Proceedings of the ICAPS Booklet on the Fifth International Planning Competition, England: IEEE, 2006: 23-25.
- [7] Hoffmann J, Nebel B, The FF planning system: fast plan generation through heuristic search [J] Journal of Artificial Intelligence Research, (2001)14: 253-302.
- [8] Gerevini A, Saetti A, Serina I. Planning through stochastic local search and temporal action graphs in lpg [J] Journal of Artificial Intelligence Research, (2003)20: 239-290.
- [9] Do M B, Kambhampati S, Sapa: A multi-objective metric temporal planner [J] Journal of Artificial Intelligence Research, (2002)20: 155-194.
- [10] Fink E, Yang Q, A spectrum of plan justifications, [C]// Proceedings of the AAAI 1993 Spring Symposium. Canada: AAAI, 1993: 23-33.
- [11] Vidal V. A lookahead strategy for heuristic search planning [C]// Proceedings of ICAPS-04. British Columbia, Canada: IEEE, 2004: 150-159.
- [12] Edelkamp S, Jabbar S, Nazih M. Large-scale optimal pddl3 planning with mips-xxl [C]// Proceedings of the ICAPS Booklet on the Fifth International Planning Competition England: IEEE, 2006: 28-30.
- [13] Hsu C W, Wah B, Huang R, et al. New features in sg-plan for handling preferences and constraints in pddl3. 0 [C]// Proceedings of the ICAPS Booklet on the Fifth International Planning Competition. USA: IEEE, 2006: 39-41.

作者简介:

谢琳女, (1970-), 硕士, 讲师. 研究方向为人工智能、机器学习、智能规划.