

一种计算动作派生前提的激活集的改进方法

蒋志华^{1),2)} 姜云飞¹⁾

¹⁾(中山大学信息科技学院软件研究所 广州 510275)

²⁾(暨南大学计算机科学系 广州 510632)

摘 要 动作的派生前提和动作删除效果的“连锁反应”是处理派生规划问题中的难点问题,基于激活集的方法是一种简单、有效的方法,但是激活集的计算时间往往过多,文中提出一种新的方法来计算激活集. LPG-td 规划系统所提出的激活集是与状态有关的并且需要在规则图上反复计算,而文中提出的激活集是与状态无关的,通过规则分裂来对规则集进行“基化”,使得寻找激活集的时间逐渐地由指数级降为线性级. 实现了一个新的能够处理派生规划问题的规划系统 LPG^{SIAS},通过对基准问题的求解,表明 LPG^{SIAS}比 LPG-td 在大部分情况下更高效. 与状态无关的激活集可以方便地转化为与状态有关的激活集,文中通过提出一种求解与状态无关的激活集的改进方法来加快对派生规划问题的求解速度.

关键词 智能规划;派生规划问题;激活集;基化

中图法分类号 TP182

An Improved Method for Calculating Activation Sets of Action Derived Preconditions

JIANG Zhi-Hua^{1),2)} JIANG Yun-Fei¹⁾

¹⁾(Software Research Institute, School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510275)

²⁾(Department of Computer Science and Technology, Jinan University, Guangzhou 510632)

Abstract The ability to deal with derived preconditions and the chain-reacting in deletion effects of actions in a derived planning domain is both practically and theoretically important. The method based on activation sets is a simple and effective way to deal with derived preconditions of actions; however, the calculation of activation sets is often time-consuming. This paper proposes a new way to calculate activation sets efficiently. Unlike the activation sets introduced first by LPG-td planner, which are closely related to the current state, activation sets that the authors define are state-independent, which means they are stable and have nothing to do with the state. In order to reduce the total time of searching activation sets, the rule set can be grounded out automatically by the technology of rule-splitting. As a result, it is possible that the complexity of calculating is reduced to linear from exponential. The proposed techniques are implemented in a new planner, called LPG^{SIAS}, and some experiments on benchmark problems show that it is more efficient than LPG-td planner in most cases. To sum up, the authors' work is to present the concept of state-independent activation sets of a derived fact and an efficient method to search activation sets so that a derived planning problem may be quickly solved.

Keywords AI planning; derived planning domains; activation sets; ground out

1 引 言

在人工智能领域,智能规划(AI Planning)问题是指从某个特定问题的初始状态出发,寻找达到解决该问题的目标状态的动作序列.国际智能规划大赛IPC(International Planning Competition)从1998年开始,每两年一次,是最有影响力的规划系统竞赛.比赛所用的规划问题描述语言PDDL(Planning Domain Description Language)总是在不断地进行扩充和增加新的特性,以测试规划系统对实际问题的近似问题的求解能力.PDDL2.2语言^[1]是2004年国际规划竞赛IPC-4所采用的规划问题描述语言,它在原来的PDDL2.1^[2]的基础上,增加了两个新的特性:派生谓词(derived predicate)和时间初始文字(time initial literal).这些特性往往被称为“Syntax Sugar”,例如,有了派生谓词,动作模型只需要描述动作的最直接效果,而事物之间千丝万缕的因果联系可以用派生规则来表示,则整个问题描述领域变得非常地简洁和清晰.定义派生谓词的规则称为派生规则,包含派生谓词的规划问题称为派生规划问题(derived planning domains).

规划问题的描述是基于—阶谓词逻辑,其中谓词分成两类:基本谓词和派生谓词^[3].二者的差别是,基本谓词可以出现在领域动作的效果中,而派生谓词不受动作的直接影响,它们在当前状态下的真值是由封闭世界假设中某些谓词通过领域规则所推导出来.派生谓词不能作为动作的直接效果,但是可以出现在动作的前提(也称为派生前提)和目标状态中.因此,如何处理动作的派生前提和动作效果的“连锁反应”是派生规划问题中的难点问题.在IPC-4上,在参加决赛的19个规划系统中,只有4个规划系统(LPG-td^[4]、SGPlan^[5]、Marvin^[6]、Downward^[7])能够求解这一类型的规划问题.可见派生谓词对大部分规划系统提出了一定难度的挑战.目前来说,对派生谓词的处理方法大致分为两类:编码方法和基于激活集的方法.编码方法主要是把派生规则转换为新的动作或者转换为条件效果,融合到已有的动作模型中.例如,Gazen和Knoblock把每条派生规则(if Φ then P)转化为一个新的动作,称为推导动作(deduction operators),规则中的 Φ 作为动作的前提条件,派生谓词 P 作为动作的增加效果^[8].Garagnani在转换为推导动作的基础上,增加了“推导事实”(deduction facts)来跟踪派生规则的

应用,推导事实记录规则应用的所有实例,以至于当规则的前提被删除时,只有确实地由该规则推导出来的派生谓词才被删除^[9].Davidson通过把新增加的谓词不断地和派生规则进行合一,使得派生规则所有可能产生的影响都编码成领域动作中的条件效果,包括条件增加效果和条件删除效果^[10].这些编码方法的共同特点是改变已有的动作模型,在预处理阶段把派生规划问题转换为等价的非派生规划问题,使得传统的规划系统可以对其进行求解.但是这类方法的问题在于引入大量的新动作或新谓词从而使动作模型变得非常复杂,问题的复杂性随着领域描述规模的增加而指数级地增长^[3],甚至某些转化过程在特殊领域中还不终止,例如,Davidson方法在Towns and Roads领域中就不能终止,因此在应用性上带来了一定的困难^[10].

而基于激活集的方法不改变已有的动作模型,而是在额外的数据结构——规则图上计算能够替代派生事实的基事实集合.基于激活集的方法是LPG-td规划系统在IPC-4上采用的方法,在处理PSR和PROMELA等两个竞赛领域都有较好的效果.LPG-td一向采用规划图(plan graph)的子图——动作图(action graph)来求解规划问题,对派生谓词的支持也是在动作图上进行扩充,形成规则动作图(rule action graph).基于激活集的方法是一种比较简单、直观、有效的方法,但是也存在不少问题,例如:派生规则要进行实例化之后产生基规则集合,基于基规则集合建立的规则图往往非常庞大,随着领域所包含的实体个数指数级地增长,在这之上计算派生事实的激活集的时间花销是值得商榷的;LPG-td所定义的激活集是与状态相关的,同一个派生事实所处的状态哪怕发生了稍微的变化,都要在规则图中重新计算其激活集;一个派生事实往往同时有多个激活集,选择一个适合当前状态的激活集,不仅可以影响求解速度还可以影响解的质量,因此关键问题是如何恰当地定义最佳激活集.而本文是对基于激活集的方法的一种改进方法,定义与状态无关的激活集并且通过规则集“基化”结合动态规划的思想来计算激活集,从而降低计算激活集的复杂性.

本文第2节介绍什么是派生规划问题,包括派生谓词的语法和语义;第3节介绍规则动作图和激活集,定义与状态无关的激活集,并指出它和与状态有关的激活集之间的关系;第4节介绍规则集的“基化”并结合动态规划的思想来计算激活集;第5节通

过实验比较 LPG^{SIAS} 和 $LPG\text{-}td$ 在求解 PSR 等基准问题上的性能;最后给出结论.

2 派生规划问题

2.1 派生谓词的语法

派生谓词不出现在动作模型中,而是由派生规则来定义在什么条件下派生谓词的真值为真. PDDL2.2 语言^[1]中对派生谓词的定義规定如下:

$\langle \text{derived-def} \rangle ::=$

$(\text{;derived } \langle \text{atomic formula}(\text{term}) \rangle \langle \text{GD} \rangle).$

其中,“; derived”是派生谓词定义的标志,“atomic formula”是原子谓词公式,即所定义的派生谓词,“GD”表示“goal description”,是形成派生规则触发条件(triggering conditions)的谓词公式.为了使规划方法容易处理派生谓词,PDDL2.2 对派生谓词的使用有如下的限制:

1) 动作不直接影响派生谓词,即派生谓词不出现在动作的任何效果中;

2) 派生谓词的真值实质上是由以下的规则推导出来:if Φ_x then $P(x)$, 其中, P 表示派生谓词, x 是变量向量,并且是 Φ 中的自由变量向量;

3) Φ 是一阶谓词公式并且满足其否定范式 NNF(Negated Normal Form),不包含任何否定形式的派生谓词.任何一个谓词公式可以通过以下“否定符号内移”的转化变成其 NNF: ① $\neg \exists x: \Phi \rightarrow \forall x: \neg \Phi$; ② $\neg \forall x: \Phi \rightarrow \exists x: \neg \Phi$; ③ $\neg \wedge \Phi_i \rightarrow \vee \neg \Phi_i$; ④ $\neg \vee \Phi_i \rightarrow \wedge \neg \Phi_i$.

派生谓词提供了一种准确而且自然的方式来表达动作的非直接效果.在一个领域所包含的所有谓词中,凡是出现在某条派生规则的头部的谓词都称为派生谓词,不是派生谓词的称为基本谓词.例如,在下图所示的 BlockWorld 领域中,谓词“on(x, y)”和“above(x, y)”的含义分别如图 1(a)和图 1(b)所示,前者是基本谓词,而后者是派生谓词,我们可以通过以下规则来推导它的实例在当前状态中的真值:

$(\text{;derived}(\text{above}?x?y))$

$(\text{or } (\text{on}?x?y)$

$(\text{exists}(?z)(\text{and } (\text{on}?x?z)$

$(\text{above}?z?y))))).$

设初始状态如图 1(c)所示,如果目标要求为“积木块 C 在积木块 A 的上方”,则我们可以表示为“above(C, A)”,但是假如没有引入派生谓词,则该目标只能表示为“on(C, A) \vee (on(C, B) \wedge on(B, A)) \vee

(on(C, D) \wedge on(D, A)) \vee (on(C, B) \wedge on(B, D) \wedge on(D, A)) \vee (on(C, D) \wedge on(D, B) \wedge on(B, A))”,可见非常繁琐.

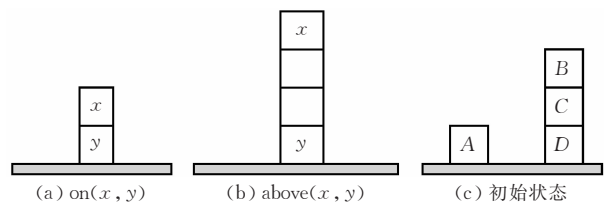


图 1 BlockWorld 领域

2.2 派生谓词的语义

粗略地说,派生谓词的一个实例(参数全部常量)为真,当且仅当它可以由某些规则派生出来,这比较类似于非单调推理.我们假设有这样的一条基规则:if B then P , P 是派生谓词的一个实例,称为派生事实(derived facts), B 是基本谓词的实例——基本事实(basic facts)组成的公式,称为该规则的触发条件.在当前状态中,如果 B 成立,则 P 也成立;而另一方面,如果由于某个动作删除了 B 中的成员,并且没有其他的规则能够派生出 P ,则 P 也被该动作删除.因此,每次产生一个新状态,我们都要对其进行扩展,使其包含在该状态下所有成立的派生谓词的实例.定义一个函数 \mathbb{D} 将基本事实集合 s 映射为在规则集 R 下的扩充集合:

$$\mathbb{D}(s) := \{s' \mid s \subseteq s', \forall (P(x), \Phi_x) \in R: \forall c, |c| = |x| : (s' \models \Phi_x) \Rightarrow P(c) \in s'\}.$$

可以看到, $\mathbb{D}(s)$ 是在规则集 R 下的闭包运算.在 PDDL2.2 对派生谓词使用的第 3 个限制条件下,即 NNF 不包含任何否定形式的派生谓词,任何一种合法的应用规则的顺序都产生相同的 $\mathbb{D}(s)$.这个结论我们可以通过一反例来说明,如果在 NNF 中允许否定形式的派生谓词,设基本事实集合是 $\{a, b, e\}$,规则集为 $\{a, b, \text{not } A \rightarrow B; a, e \rightarrow A\}$,先用第 1 条规则而后用第 2 条规则,能够推出 $\{a, b, e, B, A\}$,先用第 2 条规则而后用第 1 条规则,则只能推出 $\{a, b, e, A\}$,导致了不同的结果状态.

对于包含了派生谓词的规划问题,要对 PDDL2.1 中的某些相关的语义进行相应的修改.例如,初始状态为 s ,在应用了动作 a 之后,后继状态 s' 定义为

$$s' = \mathbb{D}(((s \setminus \bigcup_{a \in A} Del_a) \cup \bigcup_{a \in A} Add_a) \setminus D).$$

在此定义中, A 表示在某个时刻可以应用的所有动作的集合, Add_a 表示动作 a 的增加效果集合, Del_a 表示动作 a 的删除效果集合, D 是 s 中的派生事实

集合. 此定义可以确保删除那些触发条件已不成立又没有其他规则能够推出的派生事实.

包含派生谓词的规划问题称为派生规划问题, 一般地说, 一个派生规划问题是一个六元组 $\langle B, D, I, G, A, R \rangle$: B 是该领域的基本谓词集合; D 是派生谓词集合; I 表示问题的初始状态; G 表示目标状态; A 是领域动作集合; R 表示派生规则集合. 派生规划问题的解仍然是一个动作序列 $\langle a_1, a_2, \dots, a_k \rangle$,

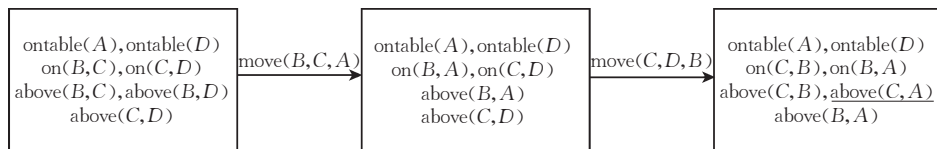


图 2 规划解对状态的改变过程

3 激活集

本文所提出的方法是对基于激活集(activation set)来求解派生规划问题的一种改进方法. 引入激活集的意义在于: 派生谓词不能出现在任何动作的效果中, 但是可以作为动作的派生前提. 在动作图上, 如果由于引入了某个动作而引入了其未被支持的派生前提, 这时不能通过增加新的动作来支持该派生前提, 只能将其转化为某些基本事实(激活集), 而这些基本事实能够通过规则集 R 将该派生前提在当前状态下推导出来. 激活集需要在规则图上进行计算, 而规则图是通过基规则集建立起来的与或图, 因此, 首先要对派生规则进行实例化以产生基规则集. 一个任意的派生规则($\text{if } \Phi_x \text{ then } P(x)$), 在一个具体的规划问题中, 可以转化成与之等效的基规则集(规则中不含任何变量). 转换过程如下: (1) Φ_x 实例化后转换为其 NNF; (2) 存在量词用所有实例的析取来代换; (3) 全称量词用所有实例的合取来代换; (4) Φ 然后被转换成析取范式 $\Phi_1 \vee \Phi_2 \vee \dots \vee \Phi_k$; (5) 对每个 Φ_i 建立一条基规则($\text{if } \Phi_i \text{ then } P$). 在本文的剩余部分, 如无特别的说明, 所说的规则集都是指基规则集.

3.1 LPG-td 所定义的状态有关的激活集

LPG-td 规划系统的求解过程是在规划空间中进行搜索, 而每个规划则是一个动作图(action graph)^[4]. 动作图 A 是必须包含 a_{start} 和 a_{end} 两个哑动作的规划图(plan graph)子图, 如果某个动作 a 位于 A 中, 则它的前提和增加效果也在 A 中.

空动作用来传播事实, 领域动作的删除效果用该领域动作和对应空动作的互斥边来表示. 线性动

通过应用它使得初始状态可以转换到目标状态, 只不过在每次应用完一个动作之后, 要对新产生的状态进行规则集 R 下的闭包运算, 使得后继动作的派生前提能够包含在新状态中. 例如, 在图 1 的示例中, 初始状态如图 1(c) 所示, 目标状态为“above(A, C)”, 动作序列“move(B, C, A); move(C, D, B)”是一个规划解, 转换过程如图 2 所示.

作图(linear action graph)的提出是为了便于求解, 如果动作图的每一个动作层至多包含一个领域动作和任意多个空动作, 则该动作图为线性动作图. 线性动作图上动作的层数并不表示执行的顺序, 不在同层的领域动作只要之间没有冲突也可以并发执行. 为了求解派生规划问题, LPG-td 定义了规则动作图(rule-action graph). 规则动作图是满足以下条件的线性动作图: (1) 每个事实层增加两种结点: 规则结点和派生结点. 规则结点用基规则的编号来标记, 派生结点用派生事实来标记; (2) 规则结点有出边和入边, 入边来自同层的形成规则条件的事实结点, 出边指向同层的由该规则所推导出的派生事实结点. 规则动作图能够很好地解决动作删除效果的“连锁反应”, 因为在规则动作图中, 规则的触发条件和所推导出的派生事实之间有规则边相连, 当动作删除了某一基本事实, 可以根据规则边来判断是否要删除该基本事实所推导的派生事实. 如果不存在其他组的规则边, 则该派生事实被删除, 而由该派生事实所推出的其他派生事实也要依次考虑. 规则动作图的方法不需要像编码方法一样引入大量的跟踪规则应用的推导事实或者条件效果. 一个规则动作图的例子如图 3 所示.

在图 3 中, 矩形结点表示动作结点, 椭圆形结点表示事实结点, 包括基本事实和派生事实, 菱形结点是规则结点. 规划问题的初始状态是 $\{p_1, p_2, p_3, p_4\}$, 目标状态是 $\{p_7, p_8\}$. 图中所示的有三条规则: $(r_1) p_3 \rightarrow d_1$, $(r_2) p_6 \rightarrow d_3$, $(r_3) d_3 \wedge p_4 \rightarrow d_4$. 对于派生前提来说, 如在本层中已经有规则推出, 则是已支持的, 反之, 是未支持的. 例如在第 3 层的 d_4 是已支持的, 第 3 层的 d_2 是未支持的. “m.e.”是互斥边, 表示动作的删除效果, 例如动作 a_2 的删除效果是 p_3 .

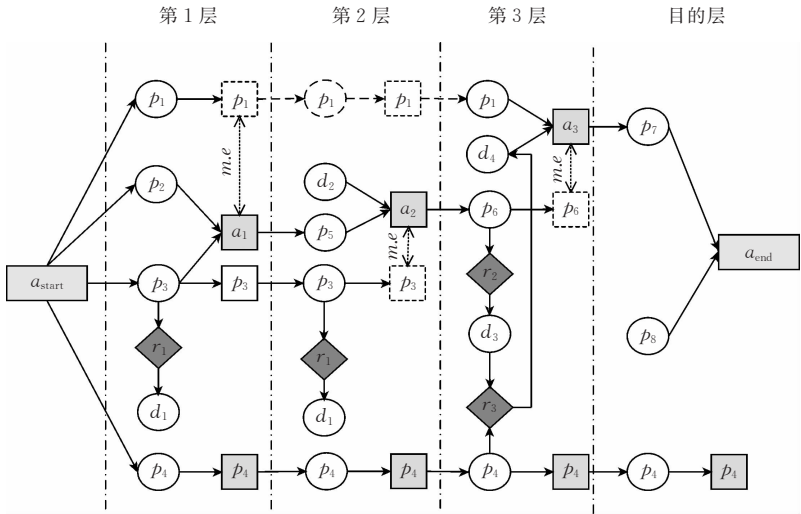


图 3 规则动作图的示例

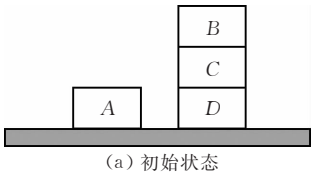
在基于激活集的方法中,动作的派生前提用其激活集来进行替换.在 LPG-td 规划系统中,对派生事实的激活集定义如下^[4].

定义 1(激活集). 在规则动作图的某一层 L 中,如果存在一个还未支持的派生前提 d ,那么 d 的激活集是一个最小的基本事实集合 F ,使得 $S(L) \cup F \vdash^R d$, $S(L)$ 表示组成层 L 的事实集合, R 是基规则集.

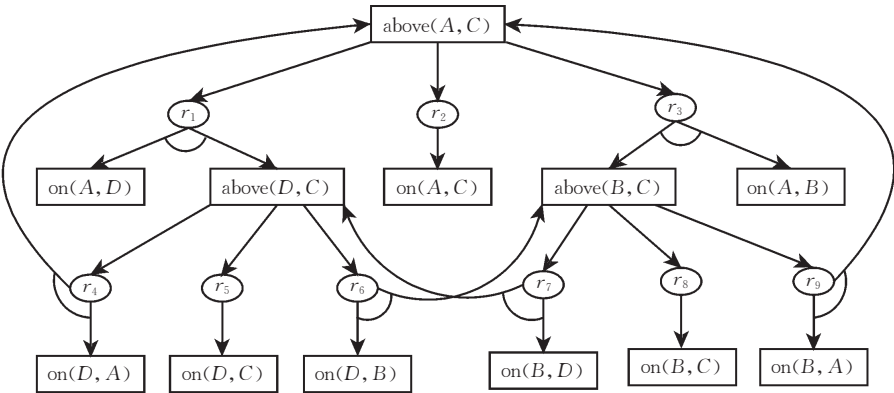
由此定义,我们可以看出,派生事实的激活集是指在当前状态已有的事实的情况下,还需要哪些基本事实来推导出该派生事实.激活集是在当前状态下未支持的基本事实集合,它引入到动作图中,通过

增加新的动作来支持它所包含的基本事实,可以产生规划空间搜索的候选动作图.一个派生事实在当前状态下可以有多个激活集,所有激活集组成一个集合,称为激活集集合 Σ .

激活集是在规则图(rule graph)上计算的.规则图是由两类结点组成的与或图(AND-OR graph):与结点是由基本事实标记的叶子结点或者由派生事实标记的内部结点,而或结点是由基规则标记的内部结点.对于一个规则结点来说,它的入边(只有一条)来自于由该规则所推导的派生事实,而它的出边(若干条)则指向该规则的所有触发条件(由基本事实或派生事实组成).在规则图上计算某个派生事实



Grounded Rules	
r1:	if on(A, D) \wedge above(D, C) then above(A, C)
r2:	if on(A, C) then above(A, C)
r3:	if on(A, B) \wedge above(B, C) then above(A, C)
r4:	if on(D, A) \wedge above(A, C) then above(D, C)
r5:	if on(D, C) then above(D, C)
r6:	if on(D, B) \wedge above(B, C) then above(D, C)
r7:	if on(B, D) \wedge above(D, C) then above(B, C)
r8:	if on(B, C) then above(B, C)
r9:	if on(B, A) \wedge above(A, C) then above(B, A)



(b) 基规则集和规则图

图 4 一个 BlockWorld 实例的规则图

的激活集,是与搜索过程和或搜索过程交替进行.在与搜索过程中,如果当前结点是一个不在当前状态的基本事实,则放到激活集集合中,如果是一个派生事实,则对它的每一个规则结点(或结点)调用或搜索过程.在或搜索过程中,对于它的每一个触发条件(与结点)进行与搜索过程.例如,在图 4(a)所示的初始状态和图 4(b)所示的规则图中,派生事实“Above(A,C)”的激活集集合是: $\Sigma = \{\{\text{on}(A,D), \text{on}(D,C)\}, \{\text{on}(A,D), \text{on}(D,B)\}, \{\text{on}(A,C)\}, \{\text{on}(A,B)\}\}$.

3.2 与状态无关的激活集

从以上激活集的定义和示例中,我们可以看出 LPG-td 所定义的激活集是与当前状态有关的.从 LPG-td 规划求解的算法上来看,在动作图的去 flaw 过程中,如果出现了派生前提,则找出其在当前状态下的所有激活集(And-Search($g, \emptyset, \emptyset, \emptyset, I \cup F$)), g 是派生事实, $I \cup F$ 表示当前状态,该函数返回激活集集合 Σ),对它们进行估值,找出一个最好的激活集($H \leftarrow \text{BestActivationSet}(\Sigma)$)来替代该派生前提($G \leftarrow G - \{g\} \cup H$).以以上例子为基础,如果我们把木块 D 和木块 C 的位置调换一下,则当前状态变为 $\{\text{table}(A), \text{table}(C), \text{on}(B,D), \text{on}(D,C)\}$,而派生事实“Above(A,C)”的激活集集合变为 $\Sigma = \{\{\text{on}(A,D)\}, \{\text{on}(A,C)\}, \{\text{on}(A,B)\}\}$.如前面所说,规则图是一个比较庞大的数据结构,随着规则实例数目的增多而指数级地增长,当动作的前提中包含较多的派生前提时,需要反复到规则图上去查找其在当前状态下的激活集集合,因此,这部分的时间开销是比较大的.为此,我们定义与状态无关的激活集,一个派生事实的与状态无关的激活集集合是唯一的,只需要在规则图中计算一次.

定义 2(与状态无关的激活集). 一个派生事实 d 的与状态无关的激活集(State-Independent Activation Set, SIAS)是一个基本事实集合 F ,使得 $F \models^R d$, R 是基规则集,并且不存在 F' ,使得 $F' \supset F$ 且 $F' \models^R d$.

我们把与状态无关的激活集集合称为 Σ^* ,在文献[11]中,我们给出了一个算法(SIAS-search($d, A, \text{Path}, \text{Open}$)),在规则图上计算与状态无关的激活集集合 Σ^* . d 是派生事实, A 是构建中的激活集, Path 用来保存已访问过的派生事实结点, Open 用来存放未访问的结点,该函数返回 Σ^* .由于规则图是有限的,该算法可以保证终止性和返回唯一的解. LPG-td 所定义的激活集是与状态相关的,相应地,我

们把它称为 State-Related Activation Set(SRAS),由这样的激活集组成的集合记为 Σ . Σ^* 可以很容易地转化为 Σ ,算法描述如下:

Algorithm(Transform(Σ^*, S)).

输入:与状态无关的激活集集合 Σ^* ,当前状态 S

输出:与状态有关的激活集集合 Σ

Begin

(1) $\Sigma \leftarrow \emptyset$

(2) For every activation set A , such that $A \in \Sigma^*$

$A' \leftarrow A - S, \Sigma \leftarrow \Sigma \cup \{A'\}$

(3) For every pair(B, B'), such that $B, B' \in \Sigma$

If $B \supseteq B'$ Then $\Sigma \leftarrow \Sigma - \{B\}$

(4) Return Σ

End.

例如,通过算法 SIAS-search($d, A, \text{Path}, \text{Open}$),我们可以得到如图 4(b)所示的规则图中派生事实“Above(A,C)”的与状态无关的激活集集合: $\Sigma^* = \{\{\text{on}(A,D), \text{on}(D,C)\}, \{\text{on}(A,D), \text{on}(D,B), \text{on}(B,C)\}, \{\text{on}(A,C)\}, \{\text{on}(A,B), \text{on}(B,D), \text{on}(D,C)\}, \{\text{on}(A,B), \text{on}(B,C)\}\}$.假设当前状态如图 4(a)所示,经过上述算法的步 2,激活集集合变为 $\{\{\text{on}(A,D), \text{on}(D,C)\}, \{\text{on}(A,D), \text{on}(D,B)\}, \{\text{on}(A,C)\}, \{\text{on}(A,B), \text{on}(B,D), \text{on}(D,C)\}, \{\text{on}(A,B)\}\}$,再经过步 3,则得到的激活集集合为 $\{\{\text{on}(A,D), \text{on}(D,C)\}, \{\text{on}(A,D), \text{on}(D,B)\}, \{\text{on}(A,C)\}, \{\text{on}(A,B)\}\}$,这与 LPG-td 通过 And-Search($g, \emptyset, \emptyset, \emptyset, I \cup F$)算法所得到的激活集集合是一样的.另外,从这个示例我们可以看出,对于同一派生事实来说, Σ^* 一般要比 Σ 大,但是它的好处在于只需要在规则图中计算一次,而当状态发生变化时, Σ 需要在规则图上重新计算.

4 规则集的“基化”

如前所述,在派生规划问题中,动作的派生前提可以用激活集来代换,但是激活集的计算时间开销不容忽视.例如,PSR(Power Supply Restoration)问题是 IPC-4 竞赛上包含派生谓词的两个基准问题之一,该问题描述的是在供电系统中,当某些电线出现故障,要重新调整电源开关给受影响的设备和线路供电的问题.这类问题所包含的动作和规则都非常多,如一个最小的实例(PSR/MIDDLE/DerivedStrips/P01_Domain. PDDL)包含 30 个基动作和 503 条基派生规则,而规模最大的实例(P50_Domain. PDDL)包括 5214 个基动作和 7450 条基派生

规则. 动作的前提中大部分也是由派生事实所构成的派生前提, 激活集的计算非常的频繁, 在规划求解的总体时间中占有不小的比例. 因此, 本文的工作是

在保留规则动作图的总体求解框架的前提下, 提出一种计算激活集的高效方法. 计算过程采用动态规划的思想, 总体框架如图 5 所示.

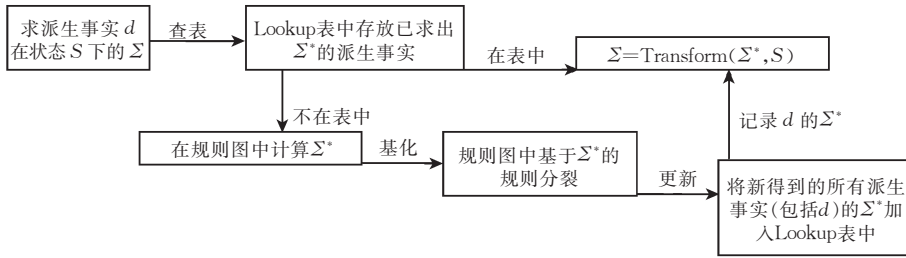


图 5 采用动态规划的过程来计算激活集

因为在给定的基规则集中, 一个派生事实与状态无关的激活集集合 Σ^* 是唯一的, 并且固定不变, 因此可以把它存在一张查找表 Lookup 中. 我们已经给出了 Σ^* 和 Σ 之间的转换关系, 在规划求解过程中, 如果要求某个派生事实 d 在状态 S 下的 Σ , 我们先检查一下 d 的 Σ^* 是否已求出. 如果已求出, 则利用算法 $\text{Transform}(\Sigma^*, S)$ 得到 Σ . 如果没有求出, 则我们在规则图上利用 $\text{SIAS-search}(d, A, \text{Path}, \text{Open})$ 过程找出 d 的 Σ^* . 在图 5 所示的总体框架中, 有一个步骤是非常重要的, 称为“基化”(ground out), 这是因为当派生事实 d 的 Σ^* 求出来之后, 通过规则集 R 下的规则推导, 可能会推出其他一些派生事实的 Σ^* , 这样我们可以顺便得到这些派生事实的 Σ^* . 把它们存储在查找表中, 如果以后要用到这些派生事实的 Σ , 我们可以直接从查找表中得知其 Σ^* , 就不需要在规则图上计算了. 在规则图上计算激活集的时间是指数级的, 在查找表中查找是线性时间的, 当求解过程进行了一段时间后, 大部分的派生事实的 Σ^* 已登记在表中, 这时整个规划的求解时间就会大大缩短了.

下面我们具体来介绍“基化”的概念和过程. “基化”过程的依据是基化前后两个规则集是等价的.

定义 3(两个规则集等价). 两个规则集 R 和 R' 是等价的, 当且仅当对于任何一个命题集合 F 和任意命题 p , 如果有 $F \vdash^R p$, 则有 $F \vdash^{R'} p$, 反过来也一样, 如果有 $F \vdash^{R'} p$, 则有 $F \vdash^R p$.

定理 1(SIAS 替换定理). 设有一命题集合 $\{p_1, p_2, \dots, p_n\}$, 每个 $p_i (i=1, 2, \dots, n)$ 都在规则集 R 的至少一条规则中出现, 如果有 $\{p_1, p_2, \dots, p_n\} \vdash^R C$, 对于规则集 R 将所有形如“ $B_1 \wedge B_2 \wedge \dots \wedge B_k \wedge C \rightarrow D$ ”的规则 r , 分别用新的规则 r' “ $B_1 \wedge B_2 \wedge \dots \wedge B_k \wedge p_1 \wedge p_2 \wedge \dots \wedge p_n \rightarrow D$ ”来进行替换, 得到一个新的规则集 R' , 则 R' 和 R 是等价的.

证明. 对于任何一个命题集合 F 和任意命题 p , 如果有 $F \vdash^R p$, 设推导路径为 $r_1, r_2, \dots, r_m, r_i \in R (i=1, 2, \dots, m)$. 分两种情况讨论: (1) 如果存在 $r_i = r (1 \leq i \leq m)$, 则 $r_i \notin R'$. 因为 $\{p_1, p_2, \dots, p_n\} \vdash^R C$, 则存在推导路径为 $r_1^*, r_2^*, \dots, r_o^*, r_j^* \in R$ 且 $r_j^* \in R' (j=1, 2, \dots, o)$, 所以原推导路径 $r_1, r_2, \dots, r_{i-1}, r_i, r_{i+1}, \dots, r_m$ 可以转化为 $r_1, r_2, \dots, r_{i-1}, r_1^*, r_2^*, \dots, r_o^*, r', r_{i+1}, \dots, r_m$, 其中 $r' \in R'$ 且 $r_j \in R' (j \neq i)$, 可得 $F \vdash^{R'} p$; (2) 如果任何 $r_i \neq r (1 \leq i \leq m)$, 则 $r_i \in R' (i=1, 2, \dots, m)$, 所以 $F \vdash^{R'} p$. 因此, 无论哪一种情况, 都可以得到结论: 如果有 $F \vdash^R p$, 则有 $F \vdash^{R'} p$.

反过来, 如果有 $F \vdash^{R'} p$, 通过与以上类似的证明过程, 可以得到 $F \vdash^R p$. 因此, 根据定义 3, R' 和 R 是等价的.

证毕.

以上定理表明, 两个规则集是等价的, 意味着在派生规划问题中将规则集进行等价的变换, 则不改变原来的规划问题.

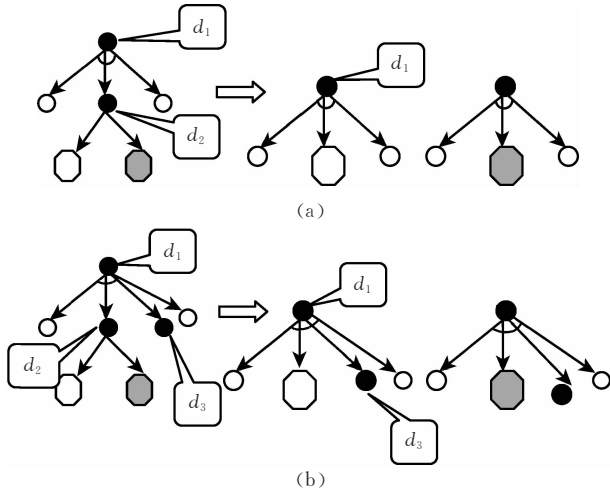
定义 4(基于派生事实 d 的 Σ^* 的规则分裂). 设派生事实 d 的 Σ^* 已知, 如下所示:

$$\left\{ \begin{array}{c} \{p_{11}, p_{12}, \dots, p_{1,k_1}\} \\ \{p_{21}, p_{22}, \dots, p_{2,k_2}\} \\ \vdots \\ \{p_{n1}, p_{n2}, \dots, p_{n,k_n}\} \end{array} \right\},$$

k_1, k_2, \dots, k_n 均为正常数, p_{ij} 均为基本事实. 对于规则集 R 中任何形如“ $B_1 \wedge B_2 \wedge \dots \wedge B_m \wedge d \rightarrow D$ ”的规则 r , 都可以产生 n 条新的规则 r_i : “ $B_1 \wedge B_2 \wedge \dots \wedge B_m \wedge p_{i1} \wedge p_{i2} \wedge \dots \wedge p_{i,k_i} \rightarrow D$ ”, $i=1, 2, \dots, n$. 这称为基于派生事实 d 的 Σ^* 的规则分裂.

根据定理 1, 基于派生事实 d 的 Σ^* 的规则分裂所产生的规则集和原来的规则集是等价的. 这样的规则分裂共有两种情况, 如图 6 所示.

图 6 是一个简化的规则图, 同一规则用与边来表示, 不同规则用或边来表示. 黑色圆圈表示派生事

图 6 基于 Σ^* 的规则分裂

实,白色圆圈表示基本事实,八角形表示激活集,为了简便起见,图中只画了两个不同的激活集,分别用白色和灰色表示.图 4(a)和(b)均表示在派生事实 d_2 的 Σ^* (共有两个激活集)求出来之后,由于派生事实 d_1 的规则条件中包含了 d_2 ,因此对 d_1 的规则进行分裂.在图 4(a)中,分裂之后 d_1 的规则中只包含基本事实,因此实际上是求出了 d_1 的两个与状态无关的激活集.而在图 4(b)中, d_1 的规则中还包含其他的派生事实 d_3 ,如果稍后 d_3 的激活集也求出来了,再进行规则分裂,我们就可以得到 d_1 的激活集.

定义 5(规则集的“基化”). 当某个派生事实 d 的 Σ_d^* 通过规则图计算得出之后,对规则集进行基于 Σ_d^* 的规则分裂.如果因此得到其他的派生事实 d_1, d_2, \dots, d_n 的激活集集合,分别记为 $\Sigma_{d_1}^*, \Sigma_{d_2}^*, \dots, \Sigma_{d_n}^*$,则对规则集进行基于 $\Sigma_{d_i}^*$ 的规则分裂 ($i=1, 2, \dots, n$).此过程不断地进行,直到不再产生新的派生事实的激活集集合为止.这个过程称为规则集的“基化”.

对规则集进行基化,其目的是使得规则的触发条件尽可能多地包含基本事实而减少派生事实,以便于激活集的计算.基化过程的算法描述如下:

Algorithm(Ground_out($DL, \Sigma_d^*, R, \text{Lookup}$)).

输入: 派生事实列表 DL, d 的与状态无关的激活集集合 Σ_d^* , 规则集 R , 查找表 Lookup

输出: 规则集 R , 查找表 Lookup

Begin

- (1) $\text{Open} \leftarrow d; \text{Open}^* \leftarrow \Sigma_d^*$
- (2) While $\text{Open} \neq \emptyset$ Do
- (3) $d \leftarrow \text{first_member}(\text{Open});$
 $\Sigma_d^* \leftarrow \text{first_member}(\text{Open}^*)$
- (4) $\text{Open} \leftarrow \text{Open} - \{d\}; \text{Open}^* \leftarrow \text{Open}^* - \{\Sigma_d^*\}$

- (5) $\text{Lookup} \leftarrow \text{Lookup} \cup \{(d, \Sigma_d^*)\}$
- (6) For every rule $r \in R$, such that $r: p_1 \wedge p_2 \wedge \dots \wedge p_k \wedge d \rightarrow C$
- (7) For every activation set $A \in \Sigma_d^*$, A is $\{a_1, a_2, \dots, a_n\}$
- (8) $r' = p_1 \wedge p_2 \wedge \dots \wedge p_k \wedge a_1 \wedge a_2 \wedge \dots \wedge a_n \rightarrow C$
- (9) $R \leftarrow R - \{r\} \cup \{r'\}$
- (10) $R \leftarrow \text{refine}(R, d);$
- (11) For every $d' \in DL$, and d' appear in R
- (12) If $\text{all_basic_precondition}(d')$ Then
- (13) $\Sigma_{d'}^* \leftarrow \text{find}(d')$
- (14) $\text{Open} \leftarrow \text{Open} \cup \{d'\}; \text{Open}^* \leftarrow \text{Open}^* \cup \{\Sigma_{d'}^*\}$

End.

在上述算法中,函数 first_member 表示取列表中的第 1 个元素.函数 refine 是对规则集进行精简,具体的功能包括:(1) 删除以 d 为结论的所有规则;(2) 删除包含不一致条件的规则;(3) 删除重复出现的规则.因为 d 的所有激活集已全部登记在 Lookup 表中,并且规则集中的 d 已被其激活集取代,所以规则集中不需要有推导 d 的规则,对应上述的功能(1).当激活集引入到规则的条件中,可能会跟原有的条件冲突,例如 Blockworld 领域中,如果规则条件中同时出现 $\text{on}(A, B)$ 和 $\text{on}(B, A)$,则为不一致的规则条件,因此需要上述功能(2).不一致的规则条件依赖于规划领域的潜在知识,可以事先设定该领域的不一致约束.例如,在 Blockworld 领域中,当规则中包含如下形式之一的条件时,则称出现了不一致的规则条件:

- i. $\text{on}(x, y) \wedge \neg \text{on}(x, y)$
- ii. $\text{on}(x1, y) \wedge \text{on}(x2, y) (x1 \neq x2)$
- iii. $\text{on}(x, y) \wedge \text{on}(y, x) (x \neq y)$
- iv. $\text{on}(x, y) \wedge \text{on}(y, z) \wedge \text{on}(z, x) (x \neq y \neq z)$
- v. ...

最后,引入的规则可能跟原有的规则重复,所以需要上述功能(3).此外,函数 $\text{all_basic_precondition}$ 是检查某个派生事实的所有规则的条件是否全为基本事实,如果全为基本事实,则调用 find 函数给出其激活集集合, find 函数只需把每个规则条件部分所包含的基本事实合并为一个激活集即可.通过“基化”过程,原先的规则集实际上分裂为两部分:精简规则集和用来存放 Σ^* 的查找表.

下面我们通过一个具体的例子来看看这样的“基化”过程.以图 4(b)中的规则集作为初始规则集,已知派生事实“above(A, C)”的 Σ^* 已求出:

$\Sigma_{\text{above}(A,C)}^* = \{\{\text{on}(A,D), \text{on}(D,C)\}, \{\text{on}(A,D), \text{on}(D,B), \text{on}(B,C)\}, \{\text{on}(A,C)\}, \{\text{on}(A,B), \text{on}(B,D), \text{on}(D,C)\}, \{\text{on}(A,B), \text{on}(B,C)\}\}$, 在经过步 6~9 和步 10 的功能(1)之后, 规则集的变化如下(变化部分用粗体表示):

$r_{4,1}$: if $\text{on}(D,A) \wedge \text{on}(A,D) \wedge \text{on}(D,C)$ then above(D,C)
 $r_{4,2}$: if $\text{on}(D,A) \wedge \text{on}(A,D) \wedge \text{on}(D,B) \wedge \text{on}(B,C)$ then above(D,C)
 $r_{4,3}$: if $\text{on}(D,A) \wedge \text{on}(A,C)$ then above(D,C)
 $r_{4,4}$: if $\text{on}(D,A) \wedge \text{on}(A,B) \wedge \text{on}(B,D) \wedge \text{on}(D,C)$ then above(D,C)
 $r_{4,5}$: if $\text{on}(D,A) \wedge \text{on}(A,B) \wedge \text{on}(B,C)$ then above(D,C)
 r_5 : if $\text{on}(D,C)$ then above(D,C)
 r_6 : if $\text{on}(D,B) \wedge \text{above}(B,C)$ then above(D,C)
 r_7 : if $\text{on}(B,D) \wedge \text{above}(D,C)$ then above(B,C)
 r_8 : if $\text{on}(B,C)$ then above(B,C)
 $r_{9,1}$: if $\text{on}(B,A) \wedge \text{on}(A,D) \wedge \text{on}(D,C)$ then above(B,C)
 $r_{9,2}$: if $\text{on}(B,A) \wedge \text{on}(A,D) \wedge \text{on}(D,B) \wedge \text{on}(B,C)$ then above(B,C)
 $r_{9,3}$: if $\text{on}(B,A) \wedge \text{on}(A,C)$ then above(B,C)
 $r_{9,4}$: if $\text{on}(B,A) \wedge \text{on}(A,B) \wedge \text{on}(B,D) \wedge \text{on}(D,C)$ then above(B,C)
 $r_{9,5}$: if $\text{on}(B,A) \wedge \text{on}(A,B) \wedge \text{on}(B,C)$ then above(B,C)

规则 r_4 和 r_9 中包含“above(A,C)”, 因此对它们要进行基于 $\Sigma_{\text{above}(A,C)}^*$ 的规则分裂. 经过步 10 的功能(2)之后, 规则集的变化如下:

$r_{4,1}$: ~~if $\text{on}(D,A) \wedge \text{on}(A,D) \wedge \text{on}(D,C)$ then above(D,C)~~
 $r_{4,2}$: ~~if $\text{on}(D,A) \wedge \text{on}(A,D) \wedge \text{on}(D,B) \wedge \text{on}(B,C)$ then above(D,C)~~
 $r_{4,3}$: if $\text{on}(D,A) \wedge \text{on}(A,C)$ then above(D,C)
 $r_{4,4}$: ~~if $\text{on}(D,A) \wedge \text{on}(A,B) \wedge \text{on}(B,D) \wedge \text{on}(D,C)$ then above(D,C)~~
 $r_{4,5}$: if $\text{on}(D,A) \wedge \text{on}(A,B) \wedge \text{on}(B,C)$ then above(D,C)
 r_5 : if $\text{on}(D,C)$ then above(D,C)
 r_6 : if $\text{on}(D,B) \wedge \text{above}(B,C)$ then above(D,C)
 r_7 : if $\text{on}(B,D) \wedge \text{above}(D,C)$ then above(B,C)
 r_8 : if $\text{on}(B,C)$ then above(B,C)
 $r_{9,1}$: if $\text{on}(B,A) \wedge \text{on}(A,D) \wedge \text{on}(D,C)$ then

above(B,C)

$r_{9,2}$: ~~if $\text{on}(B,A) \wedge \text{on}(A,D) \wedge \text{on}(D,B) \wedge \text{on}(B,C)$ then above(B,C)~~
 $r_{9,3}$: if $\text{on}(B,A) \wedge \text{on}(A,C)$ then above(B,C)
 $r_{9,4}$: ~~if $\text{on}(B,A) \wedge \text{on}(A,B) \wedge \text{on}(B,D) \wedge \text{on}(D,C)$ then above(B,C)~~
 $r_{9,5}$: ~~if $\text{on}(B,A) \wedge \text{on}(A,B) \wedge \text{on}(B,C)$ then above(B,C)~~

删除线部分是不一致的条件, 当谓词的实例不允许产生循环链时, 则出现了不一致的规则条件. 经过步 10 的功能(3)之后, 规则集的变化如下:

$r_{4,1}$: if $\text{on}(D,C)$ then above(D,C)
 $r_{4,2}$: if $\text{on}(D,B) \wedge \text{on}(B,C)$ then above(D,C)
 $r_{4,3}$: if $\text{on}(D,A) \wedge \text{on}(A,C)$ then above(D,C)
 $r_{4,5}$: if $\text{on}(D,A) \wedge \text{on}(A,B) \wedge \text{on}(B,C)$ then above(D,C)
 r_6 : if $\text{on}(D,B) \wedge \text{above}(B,C)$ then above(D,C)
 r_7 : if $\text{on}(B,D) \wedge \text{above}(D,C)$ then above(B,C)
 r_8 : if $\text{on}(B,C)$ then above(B,C)
 $r_{9,1}$: if $\text{on}(B,A) \wedge \text{on}(A,D) \wedge \text{on}(D,C)$ then above(B,C)

$r_{9,3}$: if $\text{on}(B,A) \wedge \text{on}(A,C)$ then above(B,C)
 $r_{9,4}$: if $\text{on}(B,D) \wedge \text{on}(D,C)$ then above(B,C)

此时, 查找表 *Lookup* 上登记的记录是 $\{(\text{above}(A,C), \Sigma_{\text{above}(A,C)}^*)\}$, 因为 $\text{all_basic_precondition}(\text{above}(B,C))$ 和 $\text{all_basic_precondition}(\text{above}(D,C))$ 均不为真, 因此基化过程结束. 为了更进一步说明基化过程的优点, 我们假设在随后的过程中求出了派生事实“above(B,C)”的 Σ^* 为: $\Sigma_{\text{above}(B,C)}^* = \{\{\text{on}(B,C)\}, \{\text{on}(B,A), \text{on}(A,D), \text{on}(D,C)\}, \{\text{on}(B,A), \text{on}(A,C)\}, \{\text{on}(B,D), \text{on}(D,C)\}, \{\text{on}(B,D), \text{on}(D,A), \text{on}(A,C)\}\}$, 在经过了基化过程的步 6~10 后, 规则集为

$r_{4,1}$: if $\text{on}(D,C)$ then above(D,C)
 $r_{4,2}$: if $\text{on}(D,B) \wedge \text{on}(B,C)$ then above(D,C)
 $r_{4,3}$: if $\text{on}(D,A) \wedge \text{on}(A,C)$ then above(D,C)
 $r_{4,5}$: if $\text{on}(D,A) \wedge \text{on}(A,B) \wedge \text{on}(B,C)$ then above(D,C)
 $r_{6,1}$: ~~if $\text{on}(D,B) \wedge \text{on}(B,C)$ then above(D,C)~~
 $r_{6,2}$: ~~if $\text{on}(D,B) \wedge \text{on}(B,A) \wedge \text{on}(A,D) \wedge \text{on}(D,C)$ then above(D,C)~~
 $r_{6,3}$: if $\text{on}(D,B) \wedge \text{on}(B,A) \wedge \text{on}(A,C)$ then above(D,C)

~~$r_{6,4} : \text{if on}(D, B) \wedge \text{on}(B, D) \wedge \text{on}(D, C) \text{ then above}(D, C)$~~
 ~~$r_{6,5} : \text{if on}(D, B) \wedge \text{on}(B, D) \wedge \text{on}(D, A) \wedge \text{on}(A, C) \text{ then above}(D, C)$~~

整理后为

$r_{4,1} : \text{if on}(D, C) \text{ then above}(D, C)$
 $r_{4,2} : \text{if on}(D, B) \wedge \text{on}(B, C) \text{ then above}(D, C)$
 $r_{4,3} : \text{if on}(D, A) \wedge \text{on}(A, C) \text{ then above}(D, C)$
 $r_{4,5} : \text{if on}(D, A) \wedge \text{on}(A, B) \wedge \text{on}(B, C) \text{ then above}(D, C)$
 $r_{6,3} : \text{if on}(D, B) \wedge \text{on}(B, A) \wedge \text{on}(A, C) \text{ then above}(D, C)$

可以看到, $\text{all_basic_precondition}(\text{above}(D, C))$ 为真, 因此查找表 *Lookup* 上登记的记录是 $\{(\text{above}(A, C), \Sigma^*_{\text{above}(A, C)}), (\text{above}(B, C), \Sigma^*_{\text{above}(B, C)}), (\text{above}(D, C), \Sigma^*_{\text{above}(D, C)})\}$, 其中, $\Sigma^*_{\text{above}(D, C)} = \{\{\text{on}(D, C)\}, \{\text{on}(D, B), \text{on}(B, C)\}, \{\text{on}(D, A), \text{on}(A, C)\}, \{\text{on}(D, A), \text{on}(A, B), \text{on}(B, C)\}, \{\text{on}(D, B), \text{on}(B, A), \text{on}(A, C)\}\}$. 此时, 规则集为空, 原始规则集(图 4(b))等价于此时的查找表. 以上示例表明, 在从规则图上计算出某个派生事实 d 的 Σ^*_d 后, 可以通过“基化”过程顺便找到其他派生事实 d' 的 $\Sigma^*_{d'}$, 这大大减少了查找激活集的时间.

5 实验比较

基于以上提出的方法, 我们实现了一个新的能够求解派生规划问题的规划系统 LPG^{SIAS} , 它是在 LPG1.2 规划系统^[12]上改编的. LPG1.2 规划系统参加了 IPC-3, 在自动规划系统系列中获得了杰出性能奖, 但它不能求解派生规划问题, LPG-td 是比它更新的版本, 但是尚未公布源代码. 我们将 LPG1.2 的时态动作图扩展为规则动作图, 采用 LPG-td 对候选动作图的估值方法, 然而, 在求解派生前提的激活集时采用图 5 的近似动态规划过程.

因为我们试图比较的是 LPG^{SIAS} 和 LPG-td 在其他因素几乎相同的情况下, 由于采用了不同的激活集的计算方法而导致的求解速度有何不同. 我们选取了 IPC-4 上的基准问题 (benchmark problem) 之一: $\text{PSR-Middle-StripsDerived}$ 版本下的全部问题 (50 个) 来作为测试问题, 这些问题的描述文件可以在 <http://andorfer.cs.uni-dortmund.de/~edelkamp/ipc-4/domains.html> 处下载. 由于 LPG-td 系统实际上有两个版本: $\text{LPG-speed}(\text{LPG-s})$ 和 $\text{LPG-quality}(\text{LPG-q})$, LPG-speed 是以最快速度找到一个规划解, 所找到的解往往不是最优解, 而 LPG-speed 是将找到的解不断进行迭代以找到最优解. 因此, 我们将 LPG^{SIAS} 与这两个规划系统同时进行比较, 以查看它们的求解速度和解的质量. 比较结果如表 1 所示, 本机环境为 CPU (Pentium Processor 1.2GHz) + RAM (512MB) + Redhat9.0 Linux, 编译器为 gcc 3.0.

在表 1 中, 50 个测试问题按照序号排列. “ N_{action} ”表示操作文件中动作的数目, “ N_{rule} ”表示派生规则的数目, “ N_{derived} ”表示派生事实的数目, 一般来说, 规划问题的规模越大, 所需的求解时间也越长. 我们给每个问题的限制时间是 30min, 如果在规定时间内还没有得到解, 则视为超时, 在表中用“—”来表示. “ N_{rActions} ”表示动作的个数, 因为在该领域中没有涉及到动作的代价, 因此我们用动作的个数来表示规划解的质量. 从表 1 我们可以看出, 在 90% 的情况下, LPG^{SIAS} 的求解速度要比 LPG-speed 的快, 平均减少时间为 $1008.06/50 \approx 20.17\text{s}$, 尤其像 Problem36, Problem47, Problem49 等问题, LPG^{SIAS} 的求解速度要比 LPG-speed 的快几倍. 另外, 一般来说, LPG-speed 的求解速度要比 LPG-quality 快得多, LPG-quality 的求解质量要比 LPG-speed 的好得多, 而 LPG^{SIAS} 的求解质量与 LPG-speed 的接近甚至在某些情况下要好一些, 例如 Problem29, Problem42, Problem49 等.

表 1 LPG^{SIAS} 与 LPG-td 在“ $\text{PSR-Middle-StripsDerived}$ ”领域中的比较

问题	问题规模			求解速度/s			求解质量(N_{rActions})		
	N_{action}	N_{rule}	N_{derived}	LPG-s	LPG-q	LPG^{SIAS}	LPG-s	LPG-q	LPG^{SIAS}
1	30	503	391	0.24	0.76	0.15	12	4	4
2	30	452	361	0.13	0.76	0.13	3	3	3
3	34	622	458	0.21	1.51	0.09	5	5	5
4	40	889	683	0.27	1.01	0.12	6	4	4
5	46	758	621	0.23	1.26	0.20	5	5	5
6	44	416	348	0.13	1.51	0.15	10	10	18
7	54	1823	1330	0.69	1.51	0.41	3	3	3
8	44	744	568	0.28	1.01	0.22	23	4	13

(续 表)

问题	问题规模			求解速度/s			求解质量(NrActions)		
	N_action	N_rule	N_derived	LPG-s	LPG-q	LPG ^{SIAS}	LPG-s	LPG-q	LPG ^{SIAS}
9	40	564	429	0.19	0.76	0.16	8	5	5
10	56	1233	1006	0.44	3.76	0.30	9	9	11
11	54	1128	951	0.42	1.01	0.41	6	6	6
12	82	2515	1883	1.33	2.51	1.10	9	7	7
13	74	1853	1322	0.76	2.26	1.08	11	11	17
14	84	1740	1270	0.82	1.26	0.65	6	6	6
15	88	3136	2242	1.58	9.51	0.86	12	9	9
16	86	3103	1544	0.82	1.28	0.54	7	7	7
17	90	2014	1505	1.16	1.51	0.90	6	5	5
18	86	2715	1749	1.27	12.53	1.22	10	8	8
19	136	1839	1454	0.90	3.03	0.61	12	6	8
20	142	4102	2745	5.04	314.60	3.24	39	11	23
21	152	4724	3542	3.33	7.80	2.49	13	13	13
22	138	2157	1722	1.24	6.26	1.19	18	9	11
23	144	3416	2419	2.03	4.01	1.90	13	9	18
24	146	4397	2674	2.62	3.34	2.81	3	3	3
25	246	1690	1233	1.11	12.28	0.85	26	12	22
26	254	2650	1842	1.53	313.45	0.97	19	13	16
27	252	4115	2882	3.23	51.31	2.37	17	13	16
28	256	3859	2657	2.76	3.81	2.23	11	7	10
29	274	4859	3542	11.38	272.04	9.41	102	13	23
30	270	6247	4403	16.56	31.59	12.87	15	8	16
31	260	5995	3622	11.17	105.53	5.38	13	7	7
32	272	7034	4567	34.38	358.27	18.53	64	23	27
33	526	3574	2704	2.63	5.02	2.59	12	6	12
34	516	4578	3454	4.94	14.52	4.97	32	13	31
35	528	5482	3943	10.91	29.64	5.35	18	7	7
36	532	6769	4558	122.53	320.37	11.38	10	51	8
37	542	7728	5582	17.25	63.33	14.92	12	11	12
38	540	8495	5715	108.74	—	80.75	33	91	35
39	1114	7970	5123	16.46	115.32	18.50	11	9	23
40	1106	5359	3761	358.95	319.17	203.22	52	28	35
41	1112	6530	4715	10.74	244.58	7.69	21	13	19
42	1116	7618	5099	115.21	336.70	64.03	62	13	17
43	1108	6049	4123	48.93	272.97	66.59	50	9	15
44	2406	10713	7650	138.75	357.25	115.17	8	10	8
45	2404	6265	4872	27.32	51.86	20.31	16	15	16
46	2394	8025	5680	27.99	342.00	19.14	5	5	5
47	2406	9685	6975	90.63	341.83	41.85	20	5	7
48	5210	3972	3039	20.99	314.63	28.96	11	7	17
49	5212	7632	5277	813.21	760.20	279.46	116	30	48
50	5214	7450	5177	44.51	351.76	22.53	15	12	15

求解派生规划问题.

6 结 语

本文讨论了智能规划领域中的一个重要问题——派生规划问题,并且提出了一种新的求解激活集的方法.该方法基于与状态无关的激活集的规则分裂过程,对规则集不断进行“基化”以动态地求解激活集,“基化”过程的依据在于规则集的等价变换并不改变原有规划问题的性质.基于本文所提出的方法,实现了一个新的能够求解派生规划问题的系统 LPG^{SIAS},实验结果表明,该系统能够更高效地

本文所做的工作是对如何高效地求解激活集这一论题进行探讨,但是还有一些论题尚未很好地解决.例如,如何得到派生规划问题的最优解? 如果动作的应用和推导过程再加上时间代价和资源代价,则求这类问题的最优解就变得更加复杂.激活集的选取对于规划解的质量和求解速度有直接的影响,最佳激活集可以简单地定义为能够用最少动作数从初始状态到达的激活集,然而每次选取最佳激活集是否导致最优解? 在 IPC-5 上,规划解的质量成为关注的焦点,因此在未来的工作中,我们将致力于研

究派生规划问题的最优解问题,以期望找到一种方法能够快速求解派生规划问题的最优解.

参 考 文 献

- [1] Edelkamp S, Hoffmann J. PDDL2. 2: The language for the classical part of the 4th International Planning Competition. Albert Ludwigs Universitat, Institut für Informatik, Freiburg, Germany; Technical Report 195, 2003
- [2] Fox M, Long D. PDDL2. 1: an extension to PDDL for expressing temporal planning domains. *Artificial Intelligence Research*, 2003, (20): 61-124
- [3] Thiebaux S, Hoffmann J, Nebel B. In defense of PDDL axioms. *Artificial Intelligence*, 2005, 168(1-2): 38-69
- [4] Gerevini A, Saetti A, Serina I, Toninelli P. Planning with derived predicates through Rule-Action Graphs and local search techniques. *Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence 3673*, 2005: 177-181
- [5] Wah B W, Chen Y. Partitioning of temporal planning problems in mixed space using the theory of extended saddle points. *International Journal on Artificial Intelligence Tools*, World Scientific, 2004, 13(4): 767-790
- [6] Coles A, Smith A. Marvin: Macro actions from reduced versions of the instance//*Proceedings of the Booklet of the 4th International Planning Competition*. Whistler, British, Columbia, Canada, 2004: 24-26
- [7] Helmert M, Richter S. Fast downward - making use of causal dependencies in the problem representation//Zilberstein S, Koehler J, Koenig S eds. *Proceedings of the Fourteenth International Conference on Automated Planning and Schedu-*

ling (ICAPS 2004). Whistler, British Columbia, Canada, 2004: 41-43

- [8] Gazen B C, Knoblock C A. Combining the expressivity of UCPOP with the efficiency of Graphplan//Steel S, Alami R eds. *Proceedings of the Recent Advances in AI Planning, Proceedings of the 4th European Conference on Planning (ECP'97)*. Toulouse, France, 1997: 221-233
- [9] Garagnani M. A correct algorithm for efficient planning with preprocessed domain axioms//Bramer M, Preece A, Coenen F eds. *Proceedings of the Research and Development in Intelligent Systems XVII (ES 2000)*. Berlin, 2000: 363-374
- [10] Davidson M, Garagnani M. Pre-processing planning domains containing language axioms//*Proceedings of the 21st Workshop of the UK Planning and Scheduling SIG (PlanSIG-02)*. Delft, Netherlands, 2002: 23-34
- [11] Jiang Zhi-Hua, Jiang Yun-Fei. Planning with domains containing derived predicates based on state-independent activation sets. *Computer Science*, 2007, 34(3): 176-180(in Chinese)
(蒋志华,姜云飞. 基于与状态无关的激活集的包含派生谓词的规划问题求解. *计算机科学*, 2007, 34(3): 176-180)
- [12] Gerevini A, Saetti A, Serina I. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research*, 2003, 20: 239-290
- [13] McDermott D. The planning domain definition language manual. Department of Computer Science, Yale University; Technical Report 98-003, 1998
- [14] Garagnani M. A sound linear algorithm for Pre-processing planning problems with language axioms//*Proceedings of the PLANSIG'99*. Manchester, UK, 1999: 40-53



JIANG Zhi-Hua, born in 1978, Ph.D. candidate. Her research interests include knowledge engineering and its application, AI planning and scheduling, etc.

JIANG Yun-Fei, born in 1945, professor, Ph.D. supervisor. His research interests include knowledge engineering, AI planning and intelligence diagnosis.

Background

The research work here belongs to Automated Planning, a branch of the field of Artificial Intelligence. The issue in this paper, derived planning problems, is a special type of planning problems. And unlike the general planning problems, the techniques of searching and reasoning are closely combined in solving derived planning problems. There are two primary methods for solving derived planning problems: Compiling away and activation sets based. The planning problems after compiling away can be solved by some classi-

cal planning systems; however, the plan time is increased sharply with the complexity of action models. The activation sets method was introduced first in 2003 by Gerevini et al. and implemented in a planner, called LPG-td, which joined in the International Planning Competition 2004 and outperformed other planners in the track of derived planning domains. A domain description doesn't change in the activation sets method, and activation sets for a derived fact are calculated in the rule graph and applied in the action-rule graph for

extracting solution actions. The main problem of this method is that the rule graph that is built based on the set of ground rules is often huge and calculating activation sets in it again and again is very time-consuming. In this paper, we propose an improved method for calculating activation sets efficiently. First, we present a new concept of activations sets which are state-independent, and describe the relationship between our concept and the one introduced by Gerevini et al. which is state-dependent. Then, one of the main contributions of this paper is that the rule set can be grounded out automatically by the technology of rule-splitting at plan-time, so that activation sets for some derived facts can be deduced instead of being calculated from the rule graph. The procedure of grounding out is based on the equivalence transformation of the rule set, that is to say, the original planning problem doesn't change by any means. Besides, our method is implemented in a new planner, called LPG^{SIAS} , and some experiments on benchmark problems show that it is more efficient than LPG-td planner in most cases.

Our research work is supported by the National Natural Science Foundation (60173039), and the project is aimed at

all sorts of hotspot issues and solving techniques in Automated Planning. The research team has gained a fruitful achievement in some research lines, such as non-deterministic planning, temporal planning, planning and learning, and so on. One of main trends of this field is to integrate multiple techniques together, such as searching and reasoning, and derived planning problems are an example of this trend. This paper and our earlier work (the literate [11]) have developed some creative and significant methods. The difference between these work is, that the literate [11] has presented an efficient algorithm for calculating activation sets that are state-independent in a rule graph but all the calculations are done in a preprocessing phase. However, this paper proposes a mechanism that the rule set can be grounded out automatically by the technology of rule-splitting at plan-time, so that all of activation sets are calculated at plan-time if needed, and furthermore activation sets for some derived facts can be deduced during the procedure of grounding out, instead of being calculated from the rule graph. Therefore, the method in this paper is more flexible and efficient than previous work.