CEFF——一种基于组件的条件效果处理方法

朱曼菲 姜云飞
(中山大学软件研究所 广州 510275)

摘 要 条件效果是智能规划处理更具解释性动作描述语言中最难解决的一种类型.如何扩展当前已有的规划算法,使之具有处理包含条件效果的动作描述语言的能力成为了智能规划领域中的研究热点之一.文章针对一个高效规划器 FF v2.3 在条件效果处理中存在的不足做了一些改进,提出了一个新的条件效果处理方法 CEFF.主要的改进措施有以下两点:(1)引入因子扩展法的思想将动作划分为组件,以提高对条件效果的处理效率;(2)在进行启发式估值的图扩展过程中增加对两元互斥关系的判断,以避免大部分 dead-end 状态.因此,CEFF 的实用性较 FF v2.3 要更广泛一些.

关键词 智能规划;条件效果;因子扩展法;FF;CEFF
中图法分类号 TP311

CEFF: A Method Handling Conditional Effects Based on Components

ZHU Man-Fei JIANG Yun-Fei
(*Institute of Software, Zhongshan University, Guangzhou* 510275)

Abstract Conditional Effects is the most difficult one in dealing with more expressive action description language in AI planning. It becomes one of the AI planning research's hotspots, that how to expand the existing planning algorithm to handle the action describe language with conditional effects. This paper improves FF v2.3, which is one of the most efficient planners, in handling conditional effects, and brings forward a new approach of handling conditional effects, named CEFF (Conditional Effects FF). Two main improvements of CEFF are as following: (1) adopting the soul of factor expansion, CEFF increases the efficiency of handling CE. (2) in the expanding planning graph phase, CEFF introduces the judgment of exclusive relation, and avoids the great mass of dead-end states. Therefore, CEFF is more practical than FF v2.3.

Keywords AI planning; conditional effects; factored expansion; FF; CEFF

1 引 言

近年来,智能规划的研究发展迅速,除了继续研究如何提高规划效率之外,学者们还对扩展规划算法的处理能力进行了大量的研究.目前针对扩展智能规划处理能力方面的主要的研究热点有:(1)扩展现有的规划算法,使其具有对 ADL 语言[1]的处理能力.其中 ADL 语言的描述包括了含否定式命题的前提和目标、量词效果、条件效果、析取式前提等.(2)扩展现有的规划算法,使其具有对 PDDL 语言[2]的处理能力.PDDL 语言着重于对复杂域的描述,包括带有时间、数值等因素的规划.

本文研究的就是上述第一点规划算法中对条件效果的处理方法.下文的内容安排为:首先总结和讨论已有的对包含条件效果的动作语言的处理思想;然后通过分析一个高效规划器 FF v2.3[3,4]存在的不足引出 CEFF 的思想由来;接下来系统地介绍

收稿日期:2003-06-18;修改稿收到日期:2004-09-21.**朱曼菲**,女,1977 年生,硕士,研究方向为知识工程与应用.E-mail: lncsri05@zsu.edu.cn.**姜云飞**,男,1945 年生,教授,博士生导师,研究方向为知识工程、智能规划与诊断.

CEFF 的思想、工作原理、理论证明等;最后总结全文并指出进一步的工作和展望.

2 已有的条件效果处理思想

目前对条件效果的处理思想主要有三种:完全扩展法(full expansion)[5]、IPP 使用的处理思想(简称 IPP 法)[6]以及因子扩展法(factored expansion)[7].为了更清晰地描述包含条件效果的动作语言,本节先给出一些相关的定义.

2.1 包含条件效果的动作语言

条件效果(conditional effects)[1]是指动作的一种效果,它们由蕴涵式组成,蕴涵式左边为条件部分,右边为效果部分.其形如 $C \Rightarrow E$.

也就是说,动作执行后,效果部分 E 是否成立还要取决于动作执行之前条件部分 C 是否为真,所以 C 类似于 E 的第二前提.有了条件效果的定义,我们再来定义包含了条件效果的动作语言.

包含了条件效果的动作语言[1]是指该语言所描述的动作不仅仅包含原子命题的效果,而且还包含了条件效果.

目前包含了条件效果的动作语言有 ADL[1],PDDL[2],PDDL$^+$ 等.

2.2 几种处理思想

为了便于阐明各种对包含条件结果动作语言的处理思想,本节将结合例子来讨论.例子是一个抽象的动作 A,它由前提 p 和两个效果 e 及 $(C \Rightarrow E)$ 组成,其中 $(C \Rightarrow E)$ 是条件效果,C 是条件部分,E 是效果部分.如图 1 所示.

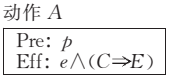

图 1 一个含有条件效果的抽象动作

下面将分别介绍三种常用的处理条件效果的思想:完全扩展法、IPP 法以及因子扩展法.

(1)完全扩展法(full expansion)

完全扩展法就是将原来包含条件效果的动作划分为多个独立的、不再含条件效果的 STRIPS[8] 动作.划分的依据是条件效果的个数以及它们所有可能的组合情况个数.下面是动作 A 的扩展例子,如图 2 所示.

这种思想的最大优点是简单.它无需改动原来的规划算法,而只需增加一个预处理阶段就扩展了原有规划算法的处理能力.但是这种思想有一个重大的缺点,在于对原有动作扩展成 STRIPS 动作时,容易导致动作数目的激增.

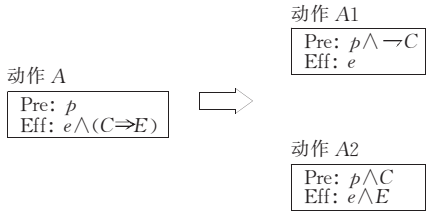

图 2 一个简单的完全扩展动作例子

(2)IPP 法

IPP 法主要思想是把所有的效果都看作是有条件的,在动作执行时,它对每个条件效果分别处理,但仍然把动作看作由图规划处理的一个基本元素,这与下面将要介绍的因子扩展法有着最大的不同.

我们可以看到例子抽象动作 A 几乎没作任何划分,只是对所有非条件效果加上空条件部分构造成一个条件效果,如图 3 所示.

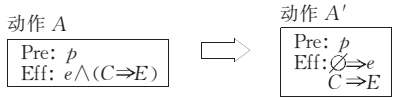

图 3 抽象动作 A 的 IPP 扩展

事实上,IPP 对条件效果的处理在于算法过程的改动,而非对动作本身结构的改动.

显而易见,IPP 法由于没有对动作结构进行重构,所以它没有增加动作的数量.与完全扩展法相比,其最大优点就在于由上述原因而带来的相对高效性(因为它不用生成指数级数量的动作).但是因为它对条件效果的处理完全依赖算法,无论是图扩展还是解抽取阶段都需要针对条件效果进行特殊的处理,所以存在增加算法复杂度的缺点.

(3)因子扩展法(factored expansion)

因子扩展法[7]的主要思想与 IPP 法有些相似,同样把所有的效果都看作条件效果,但它主要有 3 方面的改进:(1)提出了"组件"的概念,把动作按照其包含的条件效果的个数进行划分;(2)重新定义"组件"间的两元互斥关系,尤其是定义了一种新的互斥关系——导出(induce)组件互斥;(3)针对条件效果的特点重新修改了回溯的过程.由于 CEFF 采用的就是这种思想,所以其具体工作原理将在第 3 节再详细给出.

因子扩展法的优点在于提高了算法的整体效率.首先,它虽然对动作进行了划分,但这种划分仅仅由动作所含的条件效果个数来决定组件个数,这样就避免了像完全扩展法那样出现指数级数量个独

立的 STRIPS 动作的情况,所以速度能有明显的提高.另外,与 IPP 法相比,由于它采用了相对于原动作要更为独立的组件概念,使得算法能抽取更多剪枝信息,提高算法效率.

但是,因子扩展法同时也存在着缺点:复杂性加大了.首先,因为因子扩展法中每个子句得到的只是单个效果,而非整个动作的完整效果,所以在图扩展阶段定义的互斥关系需要增加更为复杂的规则;其次,在解提取阶段的回溯过程也被复杂化.这是因为需要对不希望出现的条件效果在子目标里加入它们的前提条件的否定命题等.

3　高效规划器 FF v2.3 的不足

本节先对 FF v2.3 的工作原理,尤其是对条件效果的处理思想作简要介绍;然后讨论其优缺点,并通过例子来分析它在处理条件效果上存在不足的原因所在;接下来指出论文思想的由来,并简单地介绍新方法 CEFF(Conditional Effects FF)的思想概要.

3.1　FF v2.3 的基本原理

FF v2.3 是 FF[3,4] 的一个最新版本.与原来的 FF 相比,它主要加入了对更具解释性动作描述语言的支持能力.现在先介绍 FF v2.3 的基本工作原理.

FF v2.3 主要由三部分组成:主体搜索策略、启发式值估算以及剪枝技术.其主体搜索策略采用了增强性爬山法,即爬山法和宽度优先搜索策略相结合.另外,FF v2.3 还使用了一个后备的全局搜索策略来弥补局部搜索策略的失效情况.FF v2.3 的启发式值估算使用的是放宽式图规划[9]的方法.放宽式问题即为不考虑动作删除效果的问题.FF v2.3 的最后一个重点是剪枝技术,包括“有用动作”、memorize 等技术.

在 FF v2.3 的体系中,对条件效果的处理主要体现在启发式值估算部分.FF v2.3 采用了 IPP 的思想来处理条件效果,将操作的条件效果保留至启发式值估算阶段的扩展规划图时再作实例化.具体做法为:对于给定的一个可解规划任务,规划图先将含有条件效果的操作依照当前状态层的情况(开始为初始状态层)进行例化,并将前提条件得到满足的动作加入新一层动作层中,每个动作的效果相应加入新一层状态层,这样逐层扩展,直至到达某一包含了所有目标子句的状态层.然后考察当前层的每一个目标,查找哪个条件效果的效果部分包括了考察目标,一旦找到,就将该条件效果的条件部分以及该条件效果所属的动作的前提条件都加入上一层需要实现的目标集中.当该层的目标获得者都找到后,就会在上一层生成新的目标集,如此一层一层地找,直至到达初始状态层.这样,搜索过程仅在图上作一次扫描,从最后一层找到初始层,并由此收集一个放宽式规划解$\langle A_1, A_2, \cdots, A_{m-1}\rangle$,其中 A_i 是时间步为第 i 步时的并行动作集,m 是第一次包含所有子目标的层号.

由于 FF v2.3 使用放宽式图规划来估算启发式值,所以在图扩展阶段不会有互斥关系出现.这是因为已经忽略了条件效果中的删除效果,任何动作之间都不存在冲突、不一致效果等情况.而解抽取阶段同样也是当所有目标一旦都出现在同一层命题层中时,就一定能通过一次解抽取获得放宽式图规划的一个规划解,而无需回溯.这同样是因为忽略了删除效果而取得的效果.但是,也正因为使用放宽式图规划,使得这样获得的规划解只能是估计解,有时还会离实际解相差很大,造成启发式值估算的严重偏差,带来不必要的整体效率上的浪费.下面一节将会结合一个例子来说明这一点.

3.2　FF v2.3 存在的不足

由于 FF v2.3 采用了 IPP 法的思想来处理条件效果以及利用放宽式图规划进行启发式估值,这使得它具备了总体高效性这一优点,但与此同时它又无可避免地存在因 IPP 法的特点而带来效率下降以及放宽式图规划引起的估值偏差等不足.

(1)条件效果无法预先编译带来效率的低下.

为了避免动作数量的指数级增长,FF v2.3 采用了 IPP 法的思想,不对动作进行任何拆分,于是在估算状态启发式值时就要根据放宽式规划图中当时命题层的条件满足情况来判断条件效果是否发生.这样就使得 FF v2.3 无法预先将动作实例化,从而失去了在规划图扩展时只需添加动作效果的高效性.后面可以看到,因子扩展法可以很好地解决这个不足,其原理将在论文思想概要中说明.

(2)无法避免陷入 dead-end 状态导致搜索失败.

由于 FF v2.3 使用放宽式图规划进行启发式估值,并使用无回溯的增强型爬山法作为主体搜索策略,这使 FF v2.3 的总体效率大大提高,但同时因此存在着无法避免陷入 dead-end 状态的重大不足.下面将通过一个简单例子来说明.

例 1.　假设一个规划问题的初始状态的命题

集合为$\{m\}$,目标状态的命题集合为$\{u,v\}$. 如今有如下一些可执行的动作以供选择:

动作 a: —前件 m,
—后件 Add: p, q;
动作 b: —前件 m,
—后件 Add: r;
动作 c: —前件 p,
—后件 Add:$q\rightarrow u$, Del: q;
动作 d: —前件 q,
—后件 Add:$p\rightarrow v$, Del: p;
动作 e: —前件 r,
—后件 Add: s;
动作 f: —前件 s,
—后件 Add: $r\rightarrow u$, t, Del: s;
动作 g: —前件 t,
—后件 Add: $r\rightarrow v$.

求一个合法的规划解.

现在来看看 FF v2.3 的求解过程. 由于初始状态仅包含命题 m,所以当前可执行的动作有 a,b 两个. 于是 FF v2.3 就分别为动作 A,动作 B 所生成的状态 S_a,S_b 进行启发式估值. 首先看看 S_a 的估值情况,如图 4 所示.

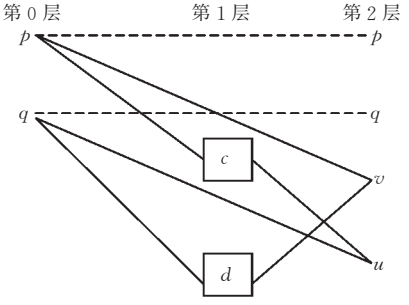

图 4 FF v2.3 对状态 S_a 的启发式值估算

由于忽略了删除效果,所以 FF v2.3 对状态 S_a 进行图扩展时只扩展了一层动作层和命题层就获得了所有的目标状态. 其解抽取阶段为每个目标选取可达的条件效果,从而找到动作 c 和动作 d 为放宽式规划解,因此,根据启发式值的估算公式,S_a 的启发式值为 2. 然而,事实上容易得知,动作 c 和动作 d 是互斥的,无论采用怎样的执行顺序都无法获得最终的目标状态,这就是 dead-end 状态,即一个可达的状态,但从它出发却无法获得目标状态. 如图 5 所示.

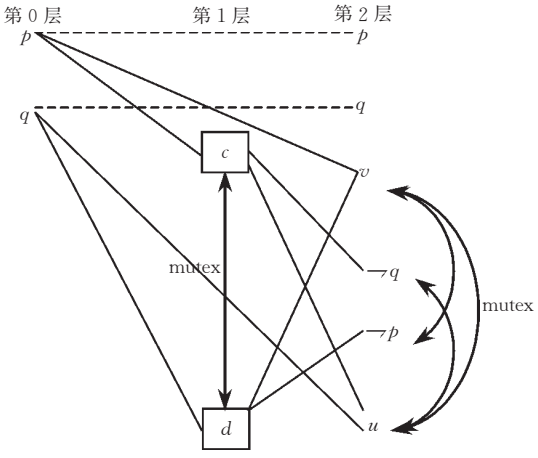

图 5 状态 S_a 的实际规划图情况

接下来看看 FF v2.3 对 S_b 的启发式值估算,如图 6 所示.

按照 FF v2.3 对该状态的解抽取情况可知,S_b 的启发式估值为 3,因为抽取的放宽式规划解包含了三个动作 e,f,g. 下面再给出 S_b 实际的图扩展情况,如图 7 所示.

从图 7 中可以看出,状态 S_b 中的目标状态命题 u 和 v 是兼容的,这时的规划图才可能存在规划解.

然而,FF v2.3 根据 S_a 的启发式值为 2,S_b 的启发式值为 3,它就会选择 S_a 进行下一步搜索. 当它发现进入状态 S_a 后没有任何动作可获得目标状态时,由于采用的是无回溯的增强型爬山法搜索策略,于是只能宣告搜索失败,并采用后备的全局搜索策略来重新求解.

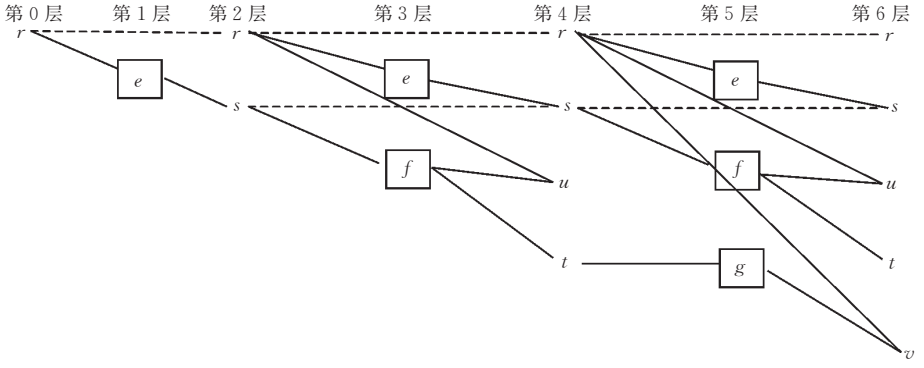

图 6 FF v2.3 对状态 S_b 的启发式值估算

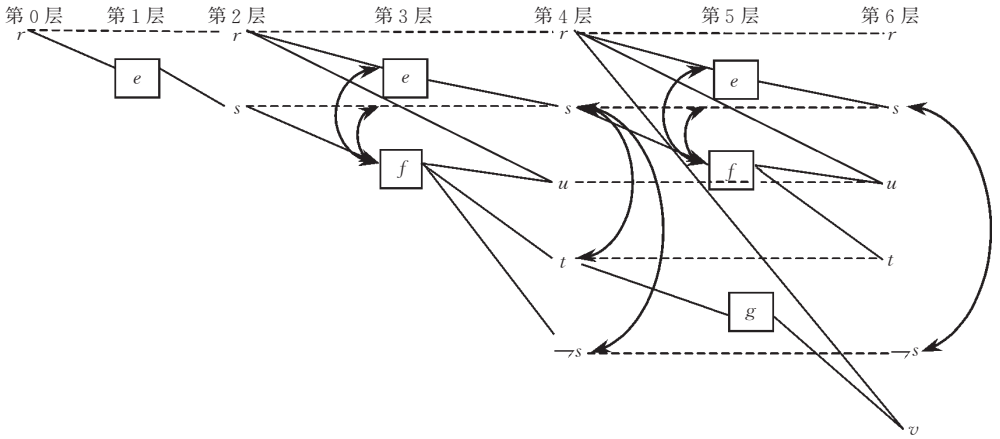

图 7　状态 S_b 的实际规划图情况

通过上面的例子可以看到,由于 FF v2.3 采用了放宽式图规划进行启发式估值,使得对动作条件效果中的删除效果部分没能体现出来,致使有些本来无法得到规划解的状态获得了较小的启发式值,令规划过程陷入了错误状态中进行搜索,并由于使用无回溯的主体搜索策略导致无法避免陷入 dead-end 状态.因为在实际领域中,dead-end 状态并不少见,但 FF v2.3 对之无法避免的这个缺点使得算法存在较大的失效性.

3.3　CEFF 的思想概要

鉴于以上对 FF v2.3 存在不足的分析,本文提出了使用一种"非完全放宽式图规划"进行启发式值估算,并采用"因子扩展法"来处理条件效果.值得说明的是,这两者并非是各自独立解决上述两个缺点的方法,而是存在有机的联系.

(1)使用"非完全放宽式图规划"进行启发式值估算.

"非完全放宽式图规划"是指在图扩展过程中仍沿用原来图规划的理论,即考虑上动作和命题之间的两元互斥关系;当满足上述的规划解存在的必要条件时才对该规划图作放宽式图规划的解抽取,即不再考虑互斥关系,只要找到一个规划解就进行启发式值估算.

之所以要在图扩展阶段考虑上互斥关系是目标状态出现在同一命题层,且两两不互斥是规划解存在的必要条件.该必要条件已由 Blum 在文献[9]中给予证明.这样就大大避免了陷入 dead-end 状态的情况①.而解抽取阶段不再考虑互斥关系是为了确保局部搜索策略的高效性.这是因为这样解抽取阶段就能通过一次回溯找到一个估算解,这与 FF v2.3 的做法相同.

(2)采用因子扩展法处理条件效果.

采用因子扩展法来处理条件效果,一来能具备直接添加效果的高效性,二来增多了对 dead-end 状态的确定,使得进一步避免了搜索失效的情形.

由于因子扩展法将动作按照条件效果划分为组件,所以可以预先完全实例化动作,从而在启发式值估算时能直接添加效果,提高对条件效果的处理效率.同时,因为因子扩展法还为组件重新定义了两元互斥关系,尤其是新增加了导出组件互斥关系,使得同层所有目标命题两两兼容时规划解存在的可能性加大了,从而尽可能避免陷入 dead-end 状态.

值得一提的是,新构造的方法仍保留 FF v2.3 的主体搜索策略,即无回溯增强型爬山法为主,最佳优先搜索为后备全局搜索策略.这是从总体的高效性考虑,若加入回溯会令启发式估值变得非常复杂,而启发式估值是需要大量计算的,所以仍采用无回溯的搜索算法.

4　一种基于组件的条件效果处理方法 CEFF

基于上面对 FF v2.3 的分析,本节针对 FF v2.3 的不足作了修改,提出一种基于组件的条件效果处理方法——CEFF(Conditional Effects FF).CEFF 的体系结构大致和 FF v2.3 相同,最大的改进是它采用了因子扩展法来处理条件效果,并使用一种基于组件的非完全放宽式图规划来估算状态的启发式值,如图 8 所示.

下面将根据 CEFF 对 FF v2.3 所作的几点改进

① dead-end 状态是难以完全避免的,因为图规划所定义的互斥关系并非是完备的,况且,Hoffman 等在文献[3]中还证明过"判断规划问题是否 dead-end free"的时间复杂度为 PSPACE.

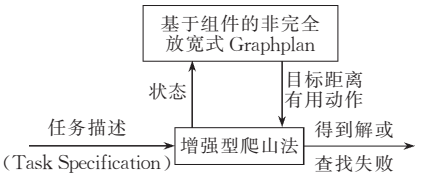

图 8 CEFF 基本体系结构图

给予详细的描述和分析,以阐明 CEFF 的工作原理.具体的内容安排为:首先给出 CEFF 的符号约定;然后介绍 CEFF 如何在预处理阶段引入组件的概念,将动作按照条件效果进行划分;接下来重构启发式值的估算方法是本节的重点,体现了 CEFF 怎样避免了 FF v2.3 对 dead-end 状态的陷入情形,并融合了因子扩展法的优点,为状态搜索过程提供了更为有效的启发式值;最后两小节都是针对基于组件后在剪枝技术和主体策略中的一些技术细节和概念上的调整.

4.1 CEFF 的符号约定

为了便于系统地介绍 CEFF 的工作原理,下面先给出 CEFF 的一些符号约定.

定义 1. 状态 S 是一个逻辑原子的有限集合.

定义 2. 包含条件效果的操作 o 是一个两元组:

$$o=(pre(o),\ ce(o)),$$

其中,$pre(o)$是动作 o 的前提条件,为原子的集合;$ce(o)$是操作 o 的条件效果,它是一个三元组 $ce^i(o)=(pre^i(o),add^i(o),del^i(o)),i\in Nat$,其中 $pre^i(o)$ 是动作 o 的第 i 个条件效果的条件部分,$add^i(o)$是 o 的第 i 个条件效果的增加效果列表,$del^i(o)$是 o 的第 i 个条件效果的删除效果列表,且三者都是原子的集合.对操作 o 进行实例化就可得到相应的动作 a.在某个状态 S 下执行一个动作 a 的结果如下:

$$Result(S,a)=\begin{cases}(S\cup A(S,a))\backslash D(S,a), & \text{若 } pre(a)\subseteq S\\ \text{不确定}, & \text{其它}\end{cases}\quad(1)$$

其中,

$$A(S,a)=\bigcup_{pre^i(a)\subseteq S} add^i(a),\ D(S,a)=\bigcup_{pre^i(a)\subseteq S} del^i(a).$$

定义 3. 组件 c^a 是一个三元组

$$c^a=(pre(c),add(c),del(c)),$$

其中,上标 a 是指该组件隶属于动作 a,$pre(c)$是组件 c^a 的前提条件,$add(c)$是 c^a 的增加效果列表,$del(c)$是 c^a 的删除效果列表,且三者都是原子的集合.并且,设组件 c 是依照操作 a 的第 i 个条件效果划分出来的组件,则满足 $pre(c)=pre(a)\cup pre^i(a)$,$add(c)=add^i(a)$,$del(c)=del^i(a)$.

定义 4. 规划任务 P 是一个三元组 $P=(O,I,G)$,其中 O 是动作的集合,I(初始状态)和 G(目标)都是原子集合.

定义 5. 给定一个规划任务 $P=(O,I,G)$,P 的基于组件规划任务 P_c 是一个三元组 $P_c=(C,I,G)$,其中 I(初始状态)和 G(目标)都是原子集合,C 是组件的集合并满足 $C=\{c^a\mid a\in A\}$,其中 A 为操作 O 的实例化动作集合.

定义 6. 给定一个基于组件规划任务 $P_c=(C,I,G)$.P_c的基于组件放宽式规划任务 P_c'被定义为 $P_c'=(C',I,G)$,其中 $C'=\{(pre(c),add(c),\phi)\mid (pre(c),add(c),del(c))\in C\}$.

定义 7. 给定一个规划任务 $P=(O,I,G)$.规划是一个动作序列 $P=\langle a_1,a_2,\cdots,a_n\rangle$,其中 $a_1,a_2,\cdots,a_n\in A$,且该动作序列能解决任务 P,其中 A 为操作 O 的实例化动作集合.

定义 8. 给定一个基于组件的放宽式规划任务 $P_c'=(C',I,G)$.基于组件的放宽式规划是一个组件序列 $P_c'=\langle c_1,c_2,\cdots,c_k\rangle$,其中 $c_1,c_2,\cdots,c_k\in C$,且该组件序列能解决放宽式任务 P'.

4.2 在预处理阶段对 ADL 语言进行转换

由于 CEFF 是在 FF v2.3 的基础上进行修改,所以仍保留 FF v2.3 对除去条件效果以外的 ADL 所包含的所有情况进行处理,并增加对动作根据其包含的条件效果的个数划分为组件的步骤.预处理阶段主要分为 6 步,如图 9 所示.

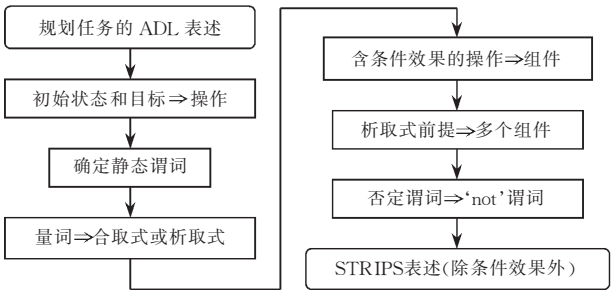

图 9 预处理阶段的六大步骤

这里仅对第 4 步进行介绍,因为这个步骤是对条件效果进行预处理的重点,其它步骤与 FF v2.3 相同.

第 4 步划分动作来生成新的组件集合主要是为 CEFF 的启发式值估算做准备的,因为 CEFF 对条件效果的处理跟 FFv2.3 一样表现在启发式值估算步骤.由于完全扩展法容易引起动作个数的指数级激增,所以不宜采用该思想.第二种思想 IPP 法正是 FF v2.3 所采用的,但前面小节也讨论过,这种思想使得 FF v2.3 不能保持 FF 原有的先实例化动

作,然后估算启发式值时可直接添加效果的高效性.于是,本文借鉴了因子扩展法对条件效果的划分思路,因为因子扩展法能较好地保持 FF 原有的高效性.

因子扩展法与 IPP 法的最大不同点在于,它把动作细化为组件,除了保留一个动作隶属号来区分组件是由不同的动作划分出来的,以及为组件引入新的互斥关系之外,因子扩展法使得规划算法能够很方便地将组件当成 STRIPS 动作来处理.当然,这仅限于图扩展阶段,对于解抽取阶段因子扩展法要复杂一些.这些将会在下一节中详细讨论,现在先来看看 CEFF 的预处理阶段是如何将动作按照条件效果划分为组件.CEFF 借鉴因子扩展法划分动作的示例如图 10 所示.

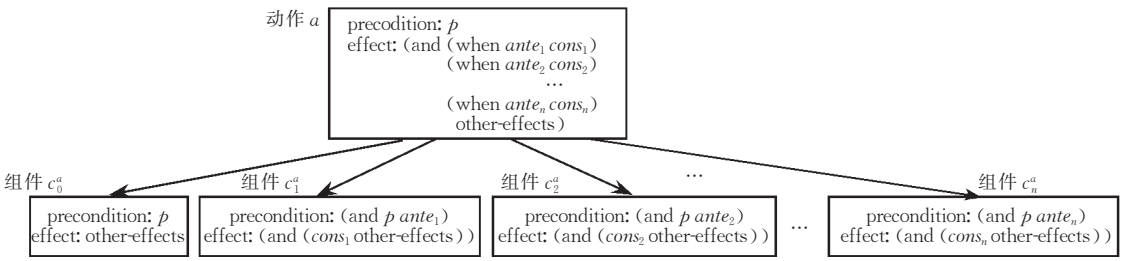

图 10　CEFF 对动作划分为组件的示例

从图中可以看出,CEFF 是按照所包含的条件效果的个数来划分动作的,因此不会引起指数级动作个数的激增.对于那些非条件效果,只需将它们当成是同一个条件部分为空的条件效果来处理就行了.具体算法如下.

算法 1.

给定:一个动作 a,它具有前提条件 p 和效果(and e_1 e_2 … e_n)以及一个新的空列表 l_c

//l_c 来记录划分动作而生成的组件

对于所有非条件效果 e_{j_1},e_{j_2},…,e_{j_k}

创建一个组件 c_0^a,其前提条件为 p,效果 ce^0 = (and e_{j_1} e_{j_2} … e_{j_k})

把 c_0^a 添加到组件列表 l_c 中

对于每一个条件效果 e_i,e_i = (when $ante_i$ $cons_i$)

创建一个组件 c_i^a,其前提条件为(and p $ante_i$),效果为(and $cons_i$ ce^0)

把 c_i^a 添加到组件列表 l_c 中

其中 c_i^a 的上标 a 就是该组件的动作隶属号.

4.3　重构启发式值的估算方法

本节将介绍 CEFF 所使用的启发式值估算方法.这里使用了一个名称"非完全放宽式图规划",这是为了和放宽式图规划区分开来.CEFF 采用了非完全放宽式图规划进行启发式值估算,即图扩展阶段要考虑上两元互斥关系,并且不满足同层目标两两兼容是不会进入解抽取阶段,从而能检测出大部分的 dead-end 状态,以运行效率换取了避免陷入无解状态.解抽取阶段则仍保留原来放宽式图规划的做法.另外,CEFF 还因采用了因子扩展法使得图规划作用的基本元素为组件,增加了互斥关系的定义,也就增多了对 dead-end 状态的剪枝,从而更有效地避免陷入无解状态.

下面将介绍基于组件的非完全放宽式图规划的工作原理.

对于一个规划任务 $P=(O,I,G)$,经过预处理阶段后生成一个新的组件集合 C,该组件集合与 I,G 一起构成了一个基于组件的规划任务 $P_c=(C,I,G)$,非完全放宽式图规划就是为其放宽式任务 P'_c 查找基于组件的放宽式规划解 $\mathcal{P}'_c=\langle C_1,C_2,\cdots,C_m\rangle$.特别的是,非完全放宽式图规划在图扩展过程中仍要考虑删除效果,这是为了更好地避免陷入无解状态而设定的.

4.3.1　图扩展过程

CEFF 对启发式值的估算也是在规划图的基础上进行的.其图扩展过程与图规划十分相似,所不同的在于这时操作的基本元素是组件,粒度比动作要小.

从初始状态开始,非完全放宽式图规划根据可供选择的组件扩展出一层组件层和一层命题层;然后根据组件和命题间的两元互斥关系标出规划图中的所有互斥结点对;接下来检查是否满足规划存在的必要条件,是,则进入解抽取阶段,否则进行新一轮的扩展,直到新的扩展对上一层的命题集合与互斥关系都没有任何改变时,即 level-off,该任务无解.其中规划存在的必要条件,即进入解抽取阶段的条件[9]是:

①目标的所有文字都出现在当前的命题层,并且

②这些文字相互之间不存在互斥关系.

下面给出图扩展过程的算法.

算法 2.

给定:规划任务 $P=(C,I,G)$,规划图 pl,且 pl 仅在第

0 层包含初始状态 I

令 $S=I, layer=0$

do

把前提已在 S 中得到满足的组件 $c_i \in C$ 加入 pl 的 $(layer+1)$层的组件层 $C_{layer+1}$ 中,并将其效果加入到 pl 的$(layer+2)$层的状态层 $S_{layer+2}$ 中

根据两元互斥关系定义新建立的组件层和状态层的互斥关系

$S=S_{layer+2}$

$layer=layer+2$

until ((S 中已包含 G,且 G 中的每两个命题不存在互斥关系)或者(S_{layer} 与 $S_{layer+2}$ 完全相同))

如果 S 中包含 G 且 G 中所有命题两两相容,则进入解抽取阶段

否则宣告算法无解

由于图扩展阶段检查了同层目标之间是否两两兼容,否则不进入解抽取阶段,所以大大减少了陷入无解状态的可能性. 接下来讨论基于组件的图扩展阶段的时间复杂度.

定理 1. 考虑一个含有 n 个操作对象,初始状态有 p 个命题以及 m 个组件的规划任务 P_c,其中每个组件具有的形式化参数个数为常数级. 令 l 为所有组件中的最大增加效果个数,则基于组件的图规划对 t 层规划图的扩展时间是 n, m, p, l 和 t 的多项式时间.

证明. 因为 m 个组件是由原来的含条件效果的动作划分而得,并且由于基于组件的规划图的图扩展过程对组件间的互斥关系重新定义了,所以整个过程执行起来实际上跟原来的图扩展是一样的,即可将 m 个组件看成是 m 个 $STRIPS$ 动作.

而对于 m 个 $STRIPS$ 动作,则文献[9]已证明图规划的图扩展过程的时间花销为 n, m, p, l 和 t 的多项式时间. 即令 k 为操作的形式化参数的最大个数. 由于操作不会新增加操作对象,所以能通过实例化使得一个操作最多能生成 $O(l\times n^k)$ 个不同的命题. 因此在规划图的同层命题层中,命题的最大个数为 $O(p+m\times l\times n^k)$. 另外,由于一个操作实例化为动作的最大个数为 $O(n^k)$,所以在规划图的同层动作层中的动作最大个数为 $O(m\times n^k)$. 而 k 为常数,因此图扩展的时间复杂度为 n, m, p, l 和 t 的多项式时间.

因此,基于组件的规划图图扩展过程的时间复杂度也是 n, m, p, l 和 t 的多项式时间. 证毕.

当满足了进入解抽取阶段的条件,即所有目标出现在同层命题层,且两两不互斥时,基于组件的非完全放宽式图规划转入解抽取阶段.

4.3.2 解抽取过程

与原来的图规划不同的是,非完全放宽式图规划在解抽取过程中不再考虑互斥关系,只需为基于组件的放宽式任务 P_c' 找到一个基于组件的放宽式规划解.

具体过程如下:从规划图的最后一层命题层开始,为每个目标查找生成该目标的组件,当该层的目标获得者都找到后,就以这些组件的前提条件集合作为上一层的目标集,如此一层一层地找,直至到达初始状态层. 由于忽略了删除效果,即不考虑互斥关系,搜索过程仅需在规划图上作一次扫描,即从最后一层找到初始层,就可以收集一个基于组件的放宽式规划解$\langle C_1, C_2, \cdots, C_{m-1}\rangle$,其中 C_i 是第 i 层组件层的并行组件集,m 是第一次包含所有子目标的层号.

下面给出该算法.

算法 3.

给定规划任务 $P=(C, I, G)$,规划图 pl,放宽式规划解 $\mathcal{P}$为空

令 max_layer 为 pl 的最大层号

令 $S_{max\_layer}=G, layer=max\_layer$

while($layer$ 不等于 0)

对S_{layer} 中的每一个 g 进行考察

如果存在 $NOOP$ 可获得 g,则选择该 $NOOP$ 为 g 的获得组件

否则按照一定的策略选取一个获得 g 的组件 c

将 c 加入规划解的第 $layer$ 层组件选取集合 C_{layer} 中

并将 c 的前提条件加入 $S_{layer-2}$

令 $layer=layer-2$

将每个 $C_i(0\leq i\leq max_layer)$中的组件根据动作隶属号合并成动作

根据动作个数为状态 I 的启发式值 $h(I)$赋值

之所以要在解抽取阶段忽略互斥关系是从 CEFF 的整体效率考虑的,因为图规划的时间花销最大是在解抽取过程,因为要找到一个合法的规划解并不容易,需要保证所选的动作和已选动作集合中的每一个都不互斥,否则就得进行重选,若都没有满足条件的,就要对上一个甚至上一层所选动作进行回溯;更甚地,若目前都无解,则要重新回到图扩展过程进行扩展,直到找到解或无解.

而 CEFF 与 FF 一样,都是通过局部搜索,对邻近的状态作启发式估值,然后选取启发式估值较优的一个状态进行转移. 这类方法有一个特点就是需

要作大量的启发式值估算,若要像图规划那样抽取解是不可能的,也不必要,因为那样一来就已经找到整个问题的最优解了.所以 CEFF 仅对基于组件的放宽式规划任务求出放宽式规划解.至于图扩展过程考虑互斥关系是为了有效避免陷入无解状态,前面已经讨论过,这里就不再重复了.

接下来看看基于组件的解抽取阶段的时间复杂度.

定理 2.　考虑一个含有 n 个操作对象,初始状态有 p 个命题以及 m 个组件的基于组件的放宽式规划任务 P'_c,其中每个组件具有的形式化参数个数为常数级.令 l 为所有组件中的最大增加效果个数,则基于组件的非完全放宽式图规划对 t 层规划图的解抽取时间是 n,m,p 和 l 的多项式时间.

证明.　解抽取过程是从规划图的最后一层开始向初始状态层查找过去,在每一层收集获得目标的组件.

令 k 为组件的形式化参数的最大个数.由于组件不会新增加操作对象,所以能通过实例化使得一个组件最多能生成 $O(l\times n^k)$ 个不同的命题.因此在规划图的同层命题层中,命题的最大个数为 $O(p+m\times l\times n^k)$.

由于不考虑互斥关系,所以为每个命题选取获得它的组件只需要常数级时间.又因为 k 为常数,所以基于组件的非完全放宽式图规划的解抽取时间复杂度是 n,m,p 和 l 的多项式时间.　　证毕.

4.3.3　启发式值估算公式

经过解抽取阶段之后,基于组件的非完全放宽式图规划得到一个基于组件的放宽式规划解 $\mathcal{P}'_c=\langle C_1,C_2,\cdots,C_{m-1}\rangle$,其中 C_i 是第 i 层组件层的并行组件集,m 是第一次包含所有子目标的层号.

由于组件只是动作根据其条件效果的个数划分出来的,组件序列也就并非是原规划问题 P 要得到的规划解.因此,需要对同层的组件根据其动作隶属号进行合并.由此,可以得到合并后的动作序列 $P'=\langle A_1,A_2,\cdots,A_{m-1}\rangle$,其中 A_i 是第 i 层动作的并行动作集,m 是第一次包含所有子目标的层号.对于一个特定状态 S,其启发式值是对串行解的长度估算,因此,启发式估算公式定义如下:

$$h(S):=\sum_{i=0,\cdots,m-1}|A_i|.$$

与 FF v2.3 相同的是,这个启发式估值也是忽略了互斥关系而抽取出来的,但保证了图扩展过程中对所有目标两两兼容,对于大部分 dead-end 状态就不会赋予启发式值,从而避免陷入其中的情形.

4.4　主体搜索策略

CEFF 与 FF v2.3 的区别主要在启发式值估算阶段,对于主体搜索策略,它们几乎是相同的.为了系统地介绍 CEFF,本小节仅简单介绍增强型爬山法的原理.

增强型爬山法就是令爬山法与宽度优先搜索法相结合,对当前状态估算其所有相邻状态的启发式值,然后从中选择最优的一个进行状态迁移.增强型爬山法是无回溯的,即一旦状态发生迁移就不再记录其它相邻的状态,即使所选状态最终会陷入无解.增强型爬山法是非完备的,为了保证算法的完备性,FF v2.3 采用了贪心法作为后备的全局搜索策略,CEFF 也保留了这种做法.

4.5　修改剪枝技术

FF v2.3 使用了一种名为"有用动作"的剪枝技术,它主要用来收集一些看起来更有利于达到规划目标的动作.具体来说,就是在启发式估值时,优先考虑那些能直接作用在当前状态下,并获得至少一个目标的动作,即有用动作.

CEFF 也采用了类似的剪枝技术,不同的是,CEFF 在启发式估值时作用的是组件,因此需要对这种技术作相应的修改,使得估值算法能优先考虑那些看起来更有利于达到目标的组件.于是,对状态 S 的有用组件的集合 $H(S)$ 定义如下:

$$H(S):=\{c\mid pre(c)\subseteq S,\exists i:pre^i(c)\subseteq S\wedge add^i(c)\cap G_1(S)\neq\varnothing\}\qquad(3)$$

其中,$G_1(S)$ 指的是由非完全放宽式 Graphplan 处理任务 (C,S,G) 时在扩展第一层命题层(非初始命题集合)时就获得的目标的集合.

同样的,有用动作在 CEFF 中也是非完备的,因为它只考虑了规划图首次扩展出来的命题层中对目标的获得情况,而没有为后继的规划给予指导性信息.因此,如果当前的有用动作给予的信息对于整个规划来说是错误的,那么,若启发式估值算法能通过图扩展的互斥关系检测到所有有用动作都无法令目标达到两两兼容,则会继续考虑其它动作;否则也会像 FF v2.3 那样陷入无解状态而宣告搜索失败,转向后备全局搜索算法来求规划解.

5　实验系统与数据

本节使用一个多仓库之间进行资源调配的应用实例来说明 CEFF 对条件效果处理的高效性以及对

无解状态的有效避免.

多仓库间的资源调配指的是这样一个实际应用问题:多个仓库原来各自存放了一些物品,为了达到一定的目的,需要对物品的存放地点作调整.如何安排运载车辆在各地仓库之间行走以及装卸库存物品以达到最终的存放格局,是规划所要解决的问题.

问题中车辆的行走就是一个典型的包含条件效果的操作,是该系统测试的重点之一.为了突出CEFF对无解状态的有效避免,实验系统还增加了对特殊物品需要指定的提货签单的限制,人为地加入无解状态.另外,该问题本来是一个既包含了条件效果又包含了时间、数值等因素的复杂域问题,但为了突出对条件效果的处理,避免令问题的重点偏移到时间、数值等方面,系统还对该问题作了必要的简化.

多仓库资源调配系统是两层体系结构的系统,前端是Java编写的GUI界面,后端是C语言编写的CEFF.操作系统平台分别是Sun Solaris 2.6,硬件平台是Sun Enterprise 450.

实验系统主要使用了两组数据进行测试.一组是不包含dead-end状态的测试数据,对比CEFF与FF v2.3在无dead-end状态下的规划效率;另一组是包含了dead-end状态的测试数据,这组测试数据主要验证FF v2.3的确会陷入无解状态,并比较CEFF与使用全局搜索策略来求解的FF v2.3的规划效率.

图11是不包含dead-end状态的测试结果,图中的横轴是问题的大小,用仓库个数和物品个数的总和来表示,纵轴是规划器的运行时间.

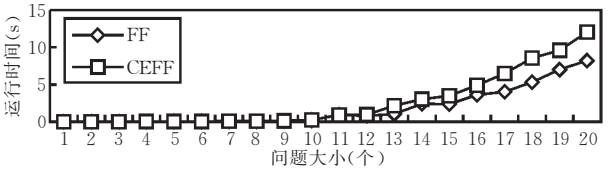

图11 无dead-end状态的CEFF和FF运行时间对比图

从图11中可以看到,CEFF在无dead-end状态的测试案例中,运行时间要比FF v2.3慢.这是因为CEFF在启发式值估算中的图扩展阶段增加了对两元互斥关系的判定,以检查当前状态往下找是否能找到解,这对于无dead-end状态的测试案例来说无疑是不起作用的.即使预处理阶段将动作划分为组件提高了规划器对条件效果的处理效率,仍无法抵消两元互斥关系判定所带来的时间花销.

但是,由于FF v2.3在含有dead-end状态的问题下有陷入无解状态的可能,并因此要转向全局搜索策略来求解,而CEFF对大部分dead-end状态都可避免,所以此时就能显示出CEFF的优势.如图11就是含有dead-end状态的测试结果.

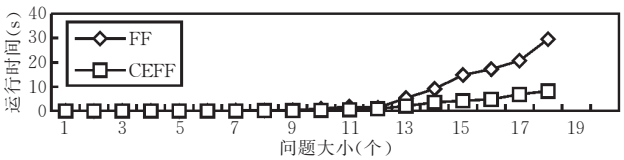

图12 含dead-end状态的CEFF和FF运行时间对比图

图11中显示了CEFF在含有dead-end状态下的运行效率要比FF v2.3高.因为此时的FF v2.3因陷入dead-end状态而无法再使用增强型爬山法作为搜索策略,而转向最佳优先的全局搜索策略,所以其运行时间也会变得很大.

通过这两组测试案例可以看到,CEFF在含有dead-end状态下的运行效率比FF v2.3高,这是采用了因子扩展法来处理条件效果,并添加了对两元互斥关系判定而取得的优势.但对于无dead-end状态的情况,CEFF有待进一步的改进.

6 总结与展望

本文介绍了一种对条件效果新的处理方法CEFF,其基本思想来源于FF v2.3,主要工作是对FF v2.3的不足之处做了两大改进:(1)引入因子扩展法的思想将动作划分为组件,以提高对条件效果的处理效率;(2)在进行启发式估值的图扩展过程增加对两元互斥关系的判断,以避免大部分dead-end状态.因此,CEFF的实用性较FF v2.3要更广泛一些,对于车辆调配、软件程序模块的配置等可能包含dead-end状态的应用领域能有更好的处理效果.

但是,如何使CEFF更好地应用于实际领域,还需要进一步地研究:

CEFF为了避免陷入dead-end状态,增加了对两元互斥关系的判断,从而使图扩展过程的性能有所下降.实现CEFF时,应该寻找良好的实现技术,提高这部分的运行性能.

把经典规划应用到实际应用领域中,除了条件效果外,还必须考虑数值、时间等因素,目前最新的经典规划通常只支持其中一个方面.由于时间关系,CEFF也仅考虑了条件效果.如何将这些因素都有机融合起来,使CEFF更加容易应用到实际应用领域中去,也是一个值得研究的问题.

CEFF 是一种通用的规划方法,如何针对特定领域加入领域知识,令 CEFF 具有更高的实用性和运行性能也是进一步的工作.

参 考 文 献

1 Pednault E. P.. ADL: Exploring the middle ground between STRIPS and the situation calculus. In: Proceedings of the 1st International Conference (KR'89), Toronto, Canada, 1989, 324～331

2 McDermott D., *et al*.. The PDDL planning domain definition language. In: Proceedings of the AIPS'98, Planning Competition Committee, Pittsburgh Pennsylvania, USA, 1998

3 Hoffmann J.. FF: The fast-forward planning system. AI Magazine, 2001, 22(3): 57～62

4 Hoffmann J., Nebel B.. The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research, 2001, 14: 253～302

5 Penberthy J. S., Weld D. S.. UCPOP: A sound, complete, partial order planner for ADL. In: Proceedings of the 3rd International Conference Principles of Knowledge Representation and Reasoning(KR'92), Cambridge, Massachusetts, USA, 1992, 103～114

6 Koehler J., Nebel B., Hoffmann J., Dimopoulos Y.. Extending planning graphs to an ADL subset. In: Proceedings of the 4th European Conference on Planning(ECP'97), 1997, 275～287

7 Anderson C. R., Smith D. E., Weld D. S.. Conditional effects in graphplan. In: Proceedings of the 4th International Conference on AI Planning Systems, Pittsburgh Pennsylvania, USA, 1998, 44～53

8 Fikes R. E., Nilsson N.. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 1971, 2: 189～208

9 Blum A. L., Furst M. L.. Fast planning through planning graph analysis. Journal of Artificial Intelligence, 1997, 90: 281～300

ZHU Man-Fei, born in 1977, M. S.. Her research interests include knowledge engineering and application.

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

Background

AI planning is one of hot branches in AI research. Given a initial state and a goal state of current problem, AI planning is to analyze the available actions and resource limitations to find out the action series which leading to the goal state. In theory research, the efficiency of planning algorithm is improved, the ability of planning algorithm about handling action describe language is extended, and new action describe language is created to describe more general problem. In practice, AI planning is applied to more and more systems, DPLAN etc. This paper focuses on expending the ability of planning algorithm, to handle the action describe language containing conditional effects.