基于模型检测的领域约束规划*

吴康恒, 姜云飞+

(中山大学 软件研究所,广东 广州 510275)

Planning with Domain Constraints Based on Model-Checking

WU Kang-Heng, JIANG Yun-Fei+

(Institute of Software, Sun Yat-Sen University, Guangzhou 510275, China)

+ Corresponding author: Phn: +86-20-84110685 ext 300, E-mail: lncsri05@zsu.edu.cn, http://soft.zsu.edu.cn

Received 2003-08-18; Accepted 2004-01-07

Wu KH, Jiang YF. Planning with domain constraints based on model-checking. *Journal of Software*, 2004, 15(11):1629~1640.

http://www.jos.org.cn/1000-9825/15/1629.htm

Abstract: The MIPS (model checking integrated planning system) has shown distinguished performance in the second and the third international planning competitions. In this paper, a declarative approach to adding domain knowledge in MIPS is presented. And DCIPS (domain constraints integrated planning system) has been developed according to this method. DCIPS allows different types of domain control knowledge such as objective, procedural or temporal knowledge to be represented and exploited in parallel, thus combining the ideas of 'planning = actions + states' into domain control knowledge. An advantage of this approach is that the domain control knowledge can be modularly formalized and added to the planning problem as desired. DCIPS is experimentally verified on the three examples in the transportation domain from AIPS 2002 planning competition where it leads to significant speed-ups.

Key words: intelligence planning; domain-dependent planning; model checking; domain constraint; transportation domain

摘 要: 基于模型检测的智能规划是当今通用的智能规划研究的热点,其求解效率比较高.但是,目前基于模型检测的智能规划系统没有考虑到利用领域知识来提高描述能力和求解效率.为此,研究了增加领域约束的基于模型检测的智能规划方法,并据此建立了基于模型检测的领域约束规划系统 DCIPS(domain constraints integrated planning system).它主要考虑了领域知识在规划中的应用,将领域知识表示为领域约束添加到规划系统中.根据"规划=动作+状态",DCIPS 将领域约束分为 3 种,即对象约束、过程约束和时序约束,采用对象约束来表达状态中对象之间的关系,采用过程约束来表达动作之间的关系,采用时序约束表达动作与状态中对象之间的关系.通过在 2002 年智能规划大赛 AIPS 2002 上关于交通运输领域的 3 个例子的测试,实验结果表明,利用领域约束的

* Supported by the National Natural Science Foundation of China under Grant No.60173039 (国家自然科学基金); the National Research Foundation for the Doctoral Program of Higher Education of China (国家教育部博士点基金)

作者简介: 吴康恒(1978－),男,广东新兴县人,博士生,主要研究领域为智能规划,机器人规划;姜云飞(1945－),男,教授,博士生导师,主要研究领域为自动推理,智能规划,模型诊断.

DCIPS 可以方便地增加领域知识,更加实用化,其效率也有了相应的提高.

关键词: 智能规划;领域依赖规划;模型检测;领域约束;交通运输

中图法分类号: TP18 **文献标识码**: A

智能规划的求解是困难的,规划问题的时间、空间复杂度都是 NP 完全的[1].目前,虽然与领域无关(domain-independent)的规划系统取得了较大的突破,例如 Graphplan[2],FF[3]和 MIPS[4],但是它们都没有利用相关的领域知识(domain-dependent knowledge)来提高规划系统的效率.

而在实际应用中,由于缺乏领域知识,这些规划系统的效率都是很差的,对稍微复杂一点的问题,就会因为空间被耗尽或时间太长而无法得出结果.因此,如何利用领域知识来增加软件系统的表达能力和求解效率就成为应用软件设计人员所面临的一个重要课题.

本文介绍了一种方法,能将领域知识方便地添加到通用的规划系统中,使得规划系统的表达能力和求解效率有所提高,又能充分利用通用的规划系统的长处.本文还描述了根据上述思想研究开发的基于模型检测的领域约束规划系统(domain constraints integrated planning system,简称 DCIPS).它考虑了领域知识在规划系统中的应用,将领域知识表示为领域约束,并根据"规划=动作+状态"的思想,将领域约束分为 3 种:对象约束、过程约束和时序约束.采用对象约束来表达状态中对象之间的关系,采用过程约束来表达动作之间的关系,采用时序约束表达动作与状态中对象之间的关系.这样的处理方法使得在解决与领域有关的问题时,DCIPS 系统比通用的规划系统显示出一些优点:(1) 加入了领域约束对领域知识进行表示,对状态可以用有层次之分的对象约束来描述,并用约束算子对它们进行操作,使得 DCIPS 的描述能力得到提高;(2) 同时通过领域约束对领域知识进行表示,对象约束、过程约束和时序约束三者相互作用来减少规划系统的搜索空间,提高 DCIPS 求解效率.通过对 2002 年智能规划大赛 AIPS 2002 上关于交通运输领域的 3 个例子进行测试,实验结果表明,利用领域约束的 DCIPS 可以方便地增加领域知识,更加实用化,其效率也有了相应的提高.

1 基于模型检测的规划

模型检测(model checking)是当前计算机研究领域中的一个热点.它将一个系统模型与逻辑需要进行比较,从而发现不一致性.传统上,这个思想用来进行硬件电路上的验证和网络协议的验证.这个思想用在智能规划中取得了较大的成功,产生了一系列功能较强的规划系统,如 MBP[5],MIPS[4],UMOP[6]等.

在智能规划研究上,为了提高规划产生的效率,研究者们根据相应的规划方法采取了不同的范式.二元判定图表 BDD(binary decision diagrams)[7]就是其中一种重要的范式.BDD 的主要优点是,采用 BDD 表达的两个谓词公式时,两个 BDD 范式逻辑相等,当且仅当这两个 BDD 范式是同一个 BDD 范式,即这两个 BDD 范式语法上相等.目前,利用 BDD 来对规划问题求解的基本思想是,先将规划问题的状态和动作表示为 BDD 范式,再将其输入到 BDD 的求解器(solver),然后将求解得到的结果转化为一般规划问题的表示.这种思想已经较多地应用在确定规划和非确定规划上.

在 2002 年 AIPS 大赛上,规划系统 MIPS(model checking integrated planning system)[4]向 PDDL2.1 扩展,能够处理数值度量和持续时间.在完全自动组,MIPS 解决问题的数量最多,是唯一一个能在所有测试例子产生规划的系统,夺得了表现优秀奖.MIPS 主要结合了模型检测技术和放宽式规划技术:采用静态分析技术对状态编码进行优化,简化了命题和实例化动作的编码空间;在象征或显式的模式数据库(pattern database,简称 PDB)内,采用放宽式规划技术(relaxed planning heuristic,简称 RPH)或加权 A*技术(BDDA*,weighted symbolic A*)对搜索过程进行估值;根据动作集和利用因果结构,采用路径分析对生成有序的规划解进行调度优化.

2 领域约束规划

当对实际的应用领域做具体的实用规划时,这些实用规划系统的规模通常是较大的、包含大量的复杂的动作集合,如果不加修改地套用通用规划的方法,其效率是很低的.但是在设计这些领域的软件时,专家们通常比

较了解的该领域的各种知识,可以用来有效地帮助软件设计人员提高应用系统的效率.有时,对领域专家来说甚至是十分简单的知识(例如,铁路运输规划中的货物必须等到车到站后才能卸车),都会对实用规划系统的效率产生重要影响.如果将这些知识添加到规划系统中,就可以提高规划系统的能力,这就是“领域依赖规划(domain-dependent planning)”的基本思想.结合以往学者们对领域规划问题的研究状况,本文分别从两方面来研究和实现领域规划问题:一方面从领域知识的表达方式和表达能力上进行研究,使得领域知识的描述能力尽量满足现实规划问题的需要,并且使规划问题的求解更加自然,以利于提高效率;另一方面,研究如何将领域知识自然地添加到目前先进的通用智能规划系统中,这样,就相当于对规划问题增加了若干有用的约束,使问题的解空间变得较小,从而提高了规划问题解决的速度和求解结果的质量,使得系统达到或接近实际应用水平.

我们在研制解决具体问题的实用规划系统时也有过上述类似的经验.为了解决某些实用规划问题,我们下载了在世界规划大赛获奖的通用规划系统,将具体的应用问题经过一定的转化后,用通用规划系统可以理解的语言 PDDL[8]进行描述,然后输入到一些现成的通用规划系统中,利用它们直接进行求解.但是,由于通用规划系统几乎没有考虑应用领域中任何有用的领域信息,系统要考虑许多在实际应用领域里根本不可能产生的状态,或在某些状态下根本不可能采取的动作,其解决问题的效率是非常低的.因此,我们将与领域密切相关的知识添加到目前世界上优秀的通用规划系统上,使它成为领域约束规划,从而能够较好地解决实际应用的问题.但是,如果这些领域知识过强,规划系统退化为一个按照固定的步骤来执行的程序,失去了普遍应用的意义,那么它也只能成为一个专门的专家系统,“有其实际的应用价值,但是缺乏研究的价值”[9].

在领域依赖规划方面,有两类方法利用领域知识,一类是层次任务分解,另一类是搜索策略控制.层次任务分解是层次规划的主要思想.它向规划系统提供一个“粗略规划(plan schemas)”,让规划系统再进行逐步的细化,这样就相当于限定了规划系统的搜索空间.这一类的代表规划系统有 UMCP 和 SHOP.搜索策略控制是基于逻辑搜索的领域依赖规划系统的基本思想.它将领域知识表达为搜索策略,并将这些搜索策略添加到与领域无关的规划系统中,对规划系统的搜索规则和启发信息进行修改,从而减少规划系统的搜索空间.这一类的代表规划系统有 TLPlan 和 TALplanner.

我们研究的 DCIPS 是基于后一种方法——搜索策略控制.下面将详细讨论其设计思路和实现方法.目前学者们对智能规划问题的看法是,给出一个任务的初始状态、目标状态和所有可行的操作,规划问题就是如何自动找到完成这一任务的动作序列.领域约束规划问题的形式化描述可以如下定义:

定义 1. 对一般规划问题的形式化定义 $M=\langle S,S_i,S_g,A,\delta\rangle$ [1]进行扩展,领域约束规划问题的形式化描述可以表示为 $M=\langle S,S_i,S_g,A,C,\delta\rangle$,其中:

S:规划系统 M 所有状态的集合;

S_i: $S_i\subseteq S$,规划系统 M 的初始状态集合;

S_g: $S_g\subseteq S$,规划系统 M 的目标状态集合;

A:规划系统 M 的有限动作集合;

C:规划系统 M 中的领域约束集合;

δ: $S\times A\to S$,规划系统 M 的动作的实例化.

在领域约束规划中,我们采用规划描述语言 PDDL 对规划问题进行描述,再对这个规划问题所属的领域抽取出相应的领域知识,这些领域知识表示为领域约束,然后将 PDDL 和领域约束输入到 DCIPS 中求解,最后得到用户要求的规划(如图 1 所示).

规划问题的形式化描述是 $M=\langle S,S_i,S_g,A,C,\delta\rangle$,采用基于这个问题描述模型来表示规划问题是比较自然的,与问题的实际情况相一致.由于把规划问题的基本要素看作状态和动作,各种要求看作状态和动作之间的各种关系,因此,规划问题很容易就转化成了一种如何将动作在状态集合中实例化的问题.这样,在模型 $M=\langle S,S_i,S_g,A,C,\delta\rangle$中,规划系统所处理的关系主要有 3 种:状态中对象之间的关系,动作集合中动作之间的关系,动作和状态中对象之间的关系.

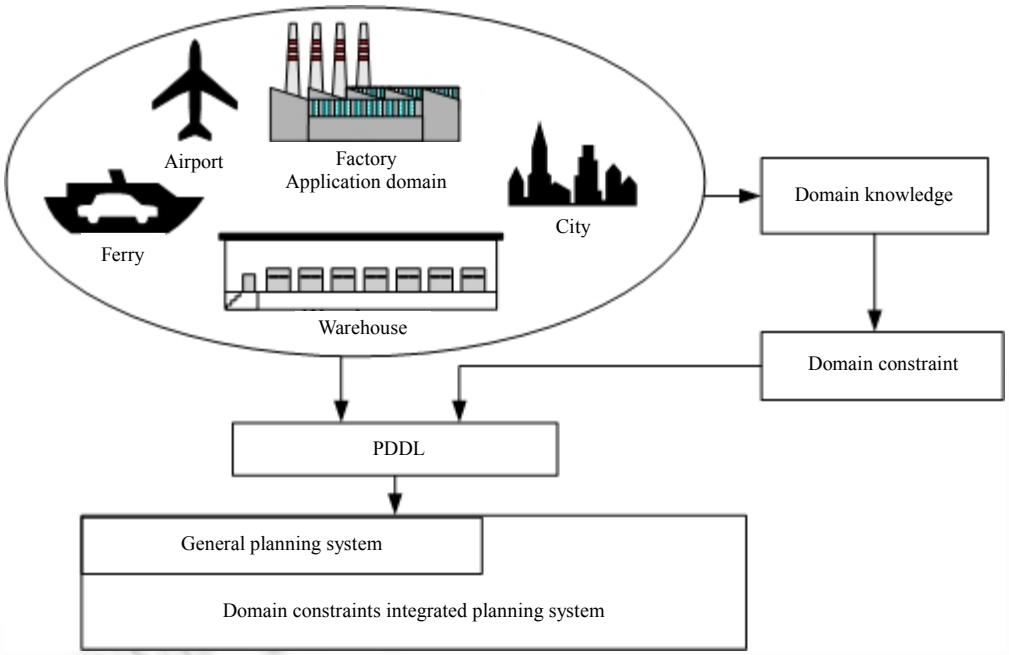

Fig.1 Design of DCIPS

图 1 领域约束规划系统设计

在“规划=动作+状态”这个模型里,利用领域知识来提高规划系统的效率和描述能力,相应地也要从这 3 种关系(对象之间、动作之间、动作和对象之间)上入手.采用领域约束来表达这 3 种关系是很自然的,因为一个约束通常是指一个包含若干变量的关系表达式,用以表示这些变量所必须满足的条件.

根据这 3 种规划系统中的关系,我们将领域约束也分为 3 种:对象约束、过程约束和时序约束.采用对象约束来表达状态中对象之间的关系,采用过程约束来表达动作之间的关系,采用时序约束来表达动作与状态中对象之间的关系,并将其添加到领域约束规划系统中.

为了使我们的领域约束规划更容易理解和比较,我们采用了 AIPS 2002 大赛上的运输领域来说明,对运输领域抽取出相关的领域知识,并且将这些领域知识表示为领域约束,再将这些领域约束添加到通用规划系统中,根据实验结果得出分析.需要强调的是,领域约束规划是对目前通用规划系统的一个改进,它不单单局限于运输领域,我们采用它作为例子只是为了使问题的描述更清晰.

2.1 DCIPS总体设计

根据上述领域约束规划的思想设计了一个规划系统 DCIPS.DCIPS 是在 MIPS 的基础上添加了领域约束的规划系统.它主要对 MIPS 的搜索模块相应地进行了修改.其主要搜索模块如图 2 所示.

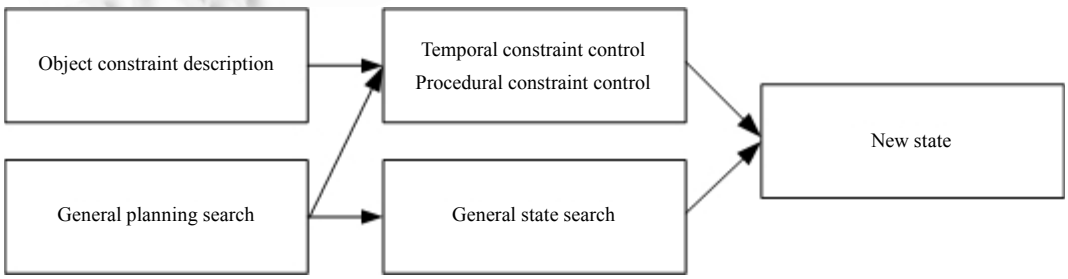

Fig.2 Search modules of DCIPS

图 2 领域约束规划搜索模块

领域约束规划搜索分为两部分:一部分是一般的状态扩展,这与通用的规划系统是一致的;另一部分是领域约束控制,使用领域约束对产生的新状态进行判断.

DCIPS 主要的算法(领域约束规划算法)描述如下:

Step 1. 初始化:输入领域描述文件 domain.pddl(状态对象、动作集合)和问题描述文件 problem.pddl(初始状态,目标状态,资源消耗最大(小)说明);输入领域约束(对象约束、过程约束、时序约束).

Step 2. 根据对象约束,预处理初始状态和目标状态.

Step 3. 将初始状态压入 Open 表.

Step 4. 取 Open 表中启发函数值 f 最小的节点 i 扩展,在扩展中考虑执行的动作是否满足过程约束和时序约束,并将已扩展的节点 i 添加到 Close 表.

Step 5. 将新扩展的节点添加到 Open 表,并计算其启发函数值 f,在计算中考虑过程约束.

Step 6. 如果节点 i 为目标状态,则成功退出,求得一个解.

2.2 对象知识约束

对于某个特定研究领域来说,存在有某些对问题中的对象与其他领域不一样的、描述能力有层次之分的领域知识,这些领域知识我们称为层次性的对象知识.DCIPS 采用约束来表达这些对象知识,并将其应用在规划系统中,称为对象知识约束,简称对象约束.

定义 2. 对象约束($OCon(x_0,x_1,...,x_n)$,Objective Constraint)是一个 $n+1$ 元谓词,其中 $x_0,x_1,...,x_n$ 是问题领域中的变元.对象约束描述了在某个领域中,变元间满足的某些该领域上的特定关系,也就是领域知识.对象约束根据作用在对象上的限定程度区分为若干的优先级的集合,这种表示就成为对象约束层次.

定义 3. 对象约束层次(OH,Objective Constraint Hierarchy)是一个根据作用在对象的限定影响来区分的领域约束有限集合,定义为

$OH_j = OH_{j-1} \cup \{OCon_0^j, OCon_1^j, ..., OCon_n^j\}$ ($j = 1,2,...$),

$OH_0 = \{OCon_0^0, OCon_1^0, ..., OCon_n^0\}$.

于是,OH_0 是对状态中对象限定影响最弱的约束集合,从 OH_0 到 OH_n 的约束强度依次增加,因此对问题领域中状态的描述能力也是从 OH_0 到 OH_n 依次增强.

定义 4. 对象约束算子(objective constraint operator,简称 OCO)是一个用来实现在同一对象约束层次上不同对象约束间的传递关系的演绎操作,定义如下:

$$OCO_1:\ \forall \alpha \quad OCon_i^j(\alpha) \to OCon_k^{j+1}(\alpha).$$

$$OCO_2:\ \forall \alpha, \forall \gamma, \text{if } (\beta_1 = \beta_2).$$

$$OCon_i^j(\alpha,\beta_1) \wedge OCon_l^{j+1}(\beta_2,\gamma) \to OCon_l^{j+1}(\alpha,\gamma).$$

$\alpha,\beta_1,\beta_2,\gamma \subseteq \{x_0,x_1,...,x_n\}$,即 α , β_1 , β_2 , γ 表示问题领域中变元的集合.

OCO_1 表示在对象约束层次 OH_{j+1} 上,由上一对象约束层次 OH_j 继承下来的一个作用在变元集 α 上的对象约束 $OCon_i^j$,可以推导出在对象约束层次 OH_{j+1} 上一个相应的对象约束 $OCon_k^{j+1}$ 也可以作用在变元集 α 上.

OCO_2 表示在对象约束层次 OH_{j+1} 上,由上一对象约束层次 OH_j 继承下来的一个作用在变元集 α 和 β_1 上的对象约束 $OCon_i^j$ 和在对象约束层次 OH_{j+1} 上作用在变元集 β_2 和 γ 上的对象约束 $OCon_l^{j+1}$,如果 $\beta_1 = \beta_2$,那么可以推导出 $OCon_l^{j+1}$ 可以作用在变元集 α , γ 上.

2.2.1 对象约束层次作为描述状态的语言

当选择某个领域进行具体研究的时候,我们会考虑如何选择一种合适的对象约束层次来描述一个可能的实际状态,这时我们关心这种对象约束层次的表达能力如何.在不同的对象约束层次上描述状态,得到的描述简洁性也不一样.

例 1:在自动仓库(depots)的例子[10]中,有 4 个集装箱(标记上 A,B,C 和 D,如图 3 所示)在桌面上,要求描述目标 g:4 个集装箱在同一柱上,其中集装箱 A 在最上面.

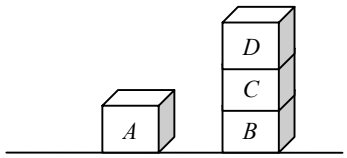

Fig.3 Four crates

图 3 4 个集装箱

为了描述这个领域,很自然可以想到的是包含对象约束 $OCon_0^0(x,y)=on(x,y)$ (表示积木 x 在积木 y 的上面,x 和 y 表面有接触)的对象约束层次 OH_0:

$$OH_0=\{on(x,y)\},$$

那么,在对象约束层次 OH_0 上,我们要把 g 表示为包含 6 个合取范式,每个范式表示一种集装箱 A,B,C 和 D 的可能的排列:

$$\begin{aligned}g=&(on(A,B)\wedge on(B,C)\wedge on(C,D))\vee\\&(on(A,B)\wedge on(B,D)\wedge on(D,C))\vee\\&(on(A,C)\wedge on(C,B)\wedge on(B,D))\vee\\&(on(A,C)\wedge on(C,D)\wedge on(D,B))\vee\\&(on(A,D)\wedge on(D,B)\wedge on(B,C))\vee\\&(on(A,D)\wedge on(D,C)\wedge on(C,B)).\end{aligned}$$

可以看出,在对象约束层次 OH_0 上的问题领域的描述是呈指数级增长的.为了避免这种情况出现,我们引入一个包含描述能力更强的对象领域约束 $OCon_0^1(x,y)=above(x,y)$ (表示积木 x 和积木 y 在同一柱上,且 x 比 y 处于更高的位置上,不要求 x 和 y 有接触)描述能力更强的对象约束层次 H_1:

$$OH_1=OH_0\cup\{above(x,y)\}=\{on(x,y),above(x,y)\},$$

那么在对象约束层次 OH_1 上,目标 g 可以简洁地表示为

$$g=above(A,B)\wedge above(A,C)\wedge above(A,D).$$

通过例 1 可以看出,将合适的对象约束层次应用在规划系统中,可以使问题领域的逻辑表示变得更为简洁、自然.

2.2.2 约束算子作为推导的演绎操作

为了实现在同一约束层次上不同约束强度的对象约束间的传递关系,我们用对象约束算子作为推导传递关系的演绎操作.

DCIPS 是把对象约束算子作为特殊的演绎操作,内嵌进来实现推导,参考了 Gazen 和 Knoblock 提出的在智能规划的预处理中解决领域公理的方法[11,12]:将推导 $p_1\wedge p_2\wedge\ldots\wedge p_n\rightarrow c$ 转化为等价的演绎操作 $(P,A,D)=([p_1\wedge p_2\wedge\ldots\wedge p_n],[c],[])$ (P:条件(premise),A:效果(affect),D:删除(delete)).那么,DCIPS 将对象约束算子 $OCon_1\wedge OCon_2\wedge\ldots\wedge OCon_n\rightarrow OCon_k$ 转化为等价的演绎操作就是 $(P,A,D)=([OCon_1\wedge OCon_2\wedge\ldots\wedge OCon_n],[OCon_k],[])$,这样就可以比较容易地将对象约束算子作为操作内嵌到规划系统中来实现求解.对象约束算子 OCO_1,OCO_2 可以表示为下面的 STRIPs 操作描述:

	OCO_1	OCO_2
P	$OCon_i^j(\alpha)$	$\beta_1=\beta_2,OCon_i^j(\alpha,\beta_1),$ $OCon_l^{j+1}(\beta_2,\gamma)$
A	$OCon_k^{j+1}(\alpha)$	$OCon_m^{j+1}(\alpha,\gamma)$
D		

例 2:实现在例 1 对象约束层次 H_1 上不同约束强度的对象约束 $on(x,y)$,$above(x,y)$ 之间的传递关系的对象约束算子 OCO_1,OCO_2.

$$OCO_1: \quad \forall x, \forall y \quad on(x,y) \rightarrow above(x,y).$$
$$OCO_2: \quad \forall x, \forall y, \forall z_1, \forall z_2 \quad \text{iff} \quad z_1 = z_2.$$
$$on(x,z_1) \wedge above(z_2,y) \rightarrow above(x,y).$$

对象约束处理算法描述如下:

Step 1. 建立对象约束和实例化对象的对应表.

Step 2. 初始化对象约束层次 OH_0.

Step 3. 使用对象约束算子从 OH_{i-1} 推导得到 OH_i,直到 $i = N$.

2.3 过程知识约束

在规划系统搜索过程中,存在一些对规划系统动作执行序列具有限定的领域知识,我们把这些领域知识称为过程领域知识.我们采用领域约束对这些过程知识进行表达,称为过程知识约束,简称过程约束(procedural constraint).过程知识大体上可以看作是大致的规划.

参考过程代数的定义[13]和过程知识在 ASP(answer set planning)中的应用[14],过程约束 pc 定义如下:

定义 5. $pc_i \Rightarrow a_i$,一个动作 a_i 可以构成一个过程约束 pc_i.

定义 6. $pc_j \Rightarrow (pc_{i1}; pc_{i2}; ...; pc_{in})$:过程约束集合的串行执行($pc_{i1}; pc_{i2}; ...; pc_{in}$)可以构成一个新的过程约束 pc_j.过程约束 pc_j 规定了 DCIPS 先执行 pc_{i1},接着 pc_{i2},…,最后执行 pc_{in}.

定义 7. $pc_j \Rightarrow (pc_{i1} \mid pc_{i2} \mid ... \mid pc_{in})$:过程约束集合的单个执行($pc_{i1} \mid pc_{i2} \mid ... \mid pc_{in}$)可以构成一个新的过程约束 pc_j.过程约束 pc_j 规定了 DCIPS 只是执行过程约束集合中 $pc_{i1}, pc_{i2}, ..., pc_{in}$ 的某个过程约束 pc_{ij}.

例如,一个过程约束 $pc = a_1; a_2; (a_3 \mid a_4 \mid a_5); a_6$, pc 表示告诉规划系统:动作 a_1 是第 1 个动作;动作 a_2 是第 2 个动作;第 3 个动作是 a_3, a_4 或者 a_5,三者中的一个;最后一个动作是 a_6.

在 DCIPS 中,过程约束用作描述在动作集合中动作与动作之间的相互关系,这里关心的是两个动作之间的先后联系.对于某个领域,可以人为地抽取出过程约束,再将其添加到 DCIPS 中.比如,在运输领域存在的相关领域知识是,“在其移动交通工具之前,通常要在其上装载着货物或刚卸载完货物,即减少交通工具的空移动”.

在自动化仓库 Depots 例子上的操作集合有 $A = \{Driver, Lift, Drop, Load, Unload\}$,得到下面的过程约束 pc:

$$pc_1 = Lift; Load; Drive; Unload; Drop,$$
$$pc_2 = Lift; Drop,$$
$$pc_3 = Drive; pc_1.$$

假设再加上卡车有着不同的速度 $Drive = \{Drive_{\text{fast}}, Drive_{\text{mid}}, Drive_{\text{low}}\}$,我们可以得到下面的过程约束:

$$pc_1 = Lift; Load; (Drive_{fast} \mid Drive_{mid} \mid Drive_{low}); Unload; Drop$$
$$pc_2 = Lift; Drop$$
$$pc_3 = (Drive_{fast} \mid Drive_{mid} \mid Drive_{low}); pc_1$$

在选择动作前判断所选动作 a 是否满足过程约束,过程约束处理算法如下:

Step 1. 建立过程约束和实例化动作的对应表.

Step 2. 取动作 a 的先辈动作序列 $preA$.

Step 3. 如果 $preA \in \{pc_1, pc_2, ..., pc_n, \}$,则 a 满足过程约束;否则, a 不满足过程约束,舍弃动作 a.

2.4 时序知识约束

时序知识容易添加到一个规划问题中,它的表示独立于动作、初始状态和目标状态的表示.采用一阶线性时序逻辑(first-order linear temporal logic,简称 LTL)来表达搜索控制用到的时序知识[15,16],这样得到了时序逻辑语言 LTL[15].LTL 以状态为可能世界,以状态的演变次序关系为可能世界间的可到达关系.在本文中讨论的时序逻辑是确定的,即状态演变次序为一个线序,所以简称线性时序逻辑为时序逻辑,它只考虑“现在状态”和“将来状态”,不考虑“过去状态”.

相对于标准一阶逻辑,LTL 主要新增加了下面 4 种时序算子(f 是 LTL 中的某个状态):

1. □称为 always 算子,□f表示“永远有 f”.

2. ◇称为 eventually 算子,◇f表示“有时有 f”.

3. ○称为 next 算子,○f表示“在下一状态(时刻)有 f”.

4. ∪称为 until 算子, $f_1 \cup f_2$ 表示“一直有 f_1,直到有 f_2”.

其中,◇时序算子在 DCIPS 中使用到,在时刻线 $M=\langle w_0,w_1,...\rangle$ 上,令 w_i 为在时间线 M 上的第 i 个时刻, V 是一个赋值函数,根据领域 D 来将 LTL 中的变量赋值,可得:

$$\langle M,w_i,V\rangle \mapsto \Diamond f_1 \text{,当且仅当,} \exists j \geq i, \langle M,w_j,V\rangle \mapsto f_1 \text{,即 } f_1 \text{ 最终会为真.}$$

2.4.1 时序约束

我们将时序知识表达为时序约束,以减少规划系统的搜索空间.时序约束定义如下:

定义 8. 时序约束(temporal constraint) $TCon(M,a,V)$ 用来表示时序知识,减少规划系统的搜索空间.令 f 是动作 a 的在产生的 end 和 overall 时刻的效果(在 PDDL2.1 第 3 层中,每个动作效果按其对状态的影响分为 3 种 start,end 和 overall),并且将来 f 不能破坏必定成立的目标状态 g, $\{\neg g\}$ 表示破坏成立的目标状态 g 的状态集合,得到时序约束:

$$(TCon(M,a,V)\mapsto f)\wedge(\Diamond f \notin \{\neg g\}).$$

在 DCIPS 中,时序约束主要用在描述动作和状态对象之间的相互关系,主要关心一个动作执行后对某个目标状态中的对象的影响.对于某个领域,人为地抽取出时序约束,再将其添加到 DCIPS 中.

例 3:例如在物流领域,根据在实际应用领域上的研究和相关文献,我们认为主要有以下几条领域知识可以提高规划系统的效率:

1. 直到需要(通常为目标位置),才从交通工具上装载或卸载货物.

2. 不要移动交通工具到不相关的位置.

3. 在移动交通工具之前,通常要在其上装载着货物或刚卸载完货物,即减少交通工具的空移动.

为了方便比较效率,我们选取在 AIPS 2002 大赛上 3 个物流领域原型的例子来研究,将上述领域知识采用时序逻辑来表达.我们主要考虑采用时序逻辑来表示其与将来必定成立的状态(通常为目标状态)相关的领域知识.令 P 表示一个包裹的名称, L_i 表示包裹 P 在初始状态的位置, L_g 表示包裹 P 在目标状态的位置, L_m 表示包裹 P 在运输途中的状态的位置.

例 4: $(TCon(M,unload(P,T,L_m),V)\mapsto at(P,L_m))\wedge(\Diamond at(P,L_m)\notin\{\neg at(P,L_g)\})$.

例 4 表示,如果包裹 P 在初始位置 L_i 一旦被装载 *load* 上运输工具(卡车) T,那么在 T 还没到达目标位置 L_g 之前,包裹 P 将不被卸载 *unload* 下运输工具 T,即一直保持在运输工具 T 上.这条领域约束就可以使得 Agent 在非目标位置时,不对包裹进行卸载.

例 5: $(TCon(M,load(P,T,L_g),V)\mapsto \neg at(P,L_g))\wedge(\Diamond\neg at(P,L_g)\notin\{\neg at(P,L_g)\})$.

例 5 表示如果包裹 P 一旦在目标位置 L_g 从运输工具(卡车) T 上卸载(unload)下来后,那么包裹 P 将不再被装载(load)到运输工具 T 上,即一直保持在目标位置 L_g 上.这条领域约束就可以使得 Agent 在包裹到达了目标位置时,不对包裹进行装载.

在自动化仓库 Depots 例子上的操作集合有 $A=\{Driver,Lift,Drop,Load,Unload\}$,令 C 表示一个集装箱的名称, P_i 表示集装箱 C 在初始状态的位置, P_g 表示集装箱 C 在目标状态的位置, P_m 表示集装箱 C 在运输途中的状态的位置.为了与原有 PDDL 说明中的 $at(x,y)$ 谓词区别,DCIPS 引入利用一个更高层的对象约束 $above(x,y)$, $above(x,y)$ 表示集装箱 x 在地点 y 之上.我们可以得到下面的时序约束:

例 6:时序约束 TC_1:

$$load_{k1}(H,C,T,P_i)\wedge(TCon(M,unload_{k2}(C,T,P_m),V)\mapsto above(C,P_m))\wedge$$
$$(\Diamond above(C,P_m)\notin\{\neg above(C,P_g)\}).$$

TC_1 表示如果集装箱 C 在时刻 k_1 初始位置 P_i 一旦被装载(load)上卡车 T,那么在 T 还没到达目标位置 P_g 之前时刻 k_2,集装箱 C 将不被卸载(unload)下卡车 T,即一直保持在卡车 T 上.这条领域约束就可以使得 DCIPS 在

非目标位置 P_m 时,不对集装箱 C 进行卸载(unload)搜索.

例 7:时序约束 TC_2 :

$$unload_{k1}(H,C,T,P_g)\wedge(TCon(M,load_{k2}(C,T,P_g),V)\mapsto\neg above(C,P_g))\wedge$$
$$(\Diamond\neg above(C,P_g)\notin\{\neg above(C,P_g)\}).$$

TC_2 表示如果集装箱 C 一旦在时刻 k_1 目标位置 P_i 从卡车 T 上卸载(unload)下来以后,集装箱 C 在时刻 k_2 就不再被装载(load)到卡车 T 上,即一直保持在目标位置 P_g 上.这条领域约束使得 DCIPS 在集装箱 C 到达目标位置 P_g 时,不对集装箱 C 进行装载(load)搜索.

在生成新状态前判断所生成的状态是否满足时序约束.具体的时序约束处理算法如下:

Step 1. 建立时序约束和实例化对象的对应表.

Step 2. 初始化对象约束.

Step 3. 如果新状态 f 满足 $(TCon(M,a,V)\mapsto f)\wedge(\Diamond f\notin\{\neg g\})$,则 f 满足时序约束;否则,f 不满足时序约束,舍弃状态 f.

3 测试和分析

3.1 实例运行结果对比

为了方便对比,我们选择在通用规划系统比赛上采用的测试用例,而且这些测试用例是在同一个领域内,可以使用相同的领域约束,不必像 TLPlan 那样再针对每个具体的测试问题手工抽取问题知识.DCIPS 添加的领域约束分别为:

1. 对象约束是 $above(x,y)$,并增加了相应的对象约束算子 OCO_1 和 OCO_2 .

$$OCO_1:\forall x,\forall y\quad on(x,y)\vee at(x,y)\rightarrow above(x,y).$$
$$OCO_2:\forall x,\forall y,\forall z_1,\forall z_2\quad \text{iff}\ \ z_1=z_2\quad on(x,z_1)\wedge above(z_2,y)\rightarrow above(x,y).$$

2. 过程约束有 pc_1 , pc_2 和 pc_3 .

$$pc_1=Lift;Load;Drive;Unload;Drop,$$
$$pc_2=Lift;Drop,$$
$$pc_3=Drive;PC_1.$$

3. 时序约束有 TC_1 和 TC_2 .

$$TC_1:load_{k1}(H,C,T,P_i)\wedge(TCon(M,unload_{k2}(C,T,P_m),V)\mapsto above(C,P_m))\wedge$$
$$(\Diamond above(C,P_m)\notin\{\neg above(C,P_g)\}).$$
$$TC_2:unload_{k1}(H,C,T,P_g)\wedge(TCon(M,load_{k2}(C,T,P_g),V)\mapsto\neg above(C,P_g))\wedge$$
$$(\Diamond\neg above(C,P_g)\notin\{\neg above(C,P_g)\}).$$

我们采用在 AIPS 2002 大赛上公布的运输领域上的 3 个测试例子(Depots,DriverLog 和 ZenoTravel)[10]作比较,其中每个比较例子我们都选取包含时序规划和度量规划的测试用例.对比规划系统我们选取了 MIPS 规划系统,MIPS 是在 AIPS 2002 大赛获得“卓越性能(distinguished performance)大奖”的 4 个规划系统之一.测试平台为 CPU(Duron 1.1G)+RAM(425M)+Redhat7.3 的 Linux 平台上运行,编译器为 gcc 2.96.下面给出测试对比结果.

因为内存容量限制,我们只能处理 Depots 问题上 pfile1~pfile4 这 4 个测试用例、DriverLog 问题上从 pfile1~pfile15 这 15 个测试用例以及 ZenoTravel 问题上 pfile1~pfile20 这 20 个测试用例.

3.1.1 Depots 例子的测试结果

在 Depots 问题上,有 6 种谓词、4 种函数和 5 种动作,DCIPS 和 MIPS 对比的测试数据和结果(运行时间、扩展节点数和最优解值)见表 1.

Table 1 Testing results comparison of Depots example

表 1 Depots 例子测试结果对比

Problem	Depots Complexity					DCIPS			MIPS		
	Depot	Truck	Pallet	Crate	Hoist	Running time (s)	Expanded nodes	Best value	Running time (s)	Expanded nodes	Best value
Pfile1	3	2	3	2	3	0.14	13	59.361 1	0.13	13	59.361 1
Pfile2	3	2	3	3	3	0.23	36	92.111 1	0.22	36	92.111 1
Pfile3	3	2	3	5	3	1.51	1 514	231.808	2.22	2 188	254.692
Pfile4	3	2	3	7	3	203.00	149 114	200.861	433.24	305 383	199.016

3.1.2 DriverLog 例子测试结果

在 DriverLog 问题上,有 5 种谓词、2 种函数和 6 种动作,DCIPS 和 MIPS 对比的测试数据和结果(运行时间、扩展节点数和最优解值)见表 2.

Table 2 Testing results comparison of Driverlog Example

表 2 DriverLog 例子测试结果对比

Problem	DriverLog Complexity				DCIPS			MIPS		
	Driver	Truck	Package	Location	Running time (s)	Expanded nodes	Best value	Running time (s)	Expanded nodes	Best value
Pfile1	2	2	2	5	0.11	8	303	0.09	8	303
Pfile2	2	2	3	7	0.13	29	310	0.13	32	310
Pfile3	2	2	4	6	0.13	13	173	0.13	13	173
Pfile4	3	2	4	7	0.15	21	392	0.15	20	392
Pfile5	3	2	5	6	0.17	31	112	0.17	31	112
Pfile6	2	3	6	18	0.20	27	260	0.20	19	260
Pfile7	2	3	6	28	0.21	13	268	0.21	13	268
Pfile8	2	3	6	37	0.77	868	318	0.46	412	313
Pfile9	3	3	6	25	0.39	53	870	0.38	64	980
Pfile10	4	4	8	35	0.59	26	340	0.58	24	340
Pfile11	5	5	10	43	0.80	36	391	0.80	36	391
Pfile12	5	5	15	48	13.01	3893	486	3.58	700	611
Pfile13	5	5	20	54	2.51	48	563	2.55	51	558
Pfile14	5	5	25	59	5.30	846	888	4.88	763	1 049
Pfile15	8	6	25	59	20.06	365	714	27.54	712	893

3.1.3 ZenoTravel 例子测试结果

在 ZenoTravel 问题上,有 2 种谓词、11 种函数和 5 种动作,DCIPS 和 MIPS 对比的测试数据和结果(运行时间、扩展节点数和最优解值)见表 3.

Table 3 Testing results comparison of ZenoTravel example

表 3 ZenoTravel 例子测试结果对比

Problem	ZenoTravel Complexity			DCIPS			MIPS		
	Aircraft	Person	City	Running time (s)	Expanded nodes	Best value	Running time (s)	Expanded nodes	Best value
Pfile1	1	2	3	0.10	1	27.257	0.10	1	27.257
Pfile2	1	3	3	0.09	7	30.210 4	0.09	7	30.210 4
Pfile3	2	4	3	0.12	15	18.152 7	0.12	15	18.152 7
Pfile4	2	5	3	0.13	19	153.294	0.13	19	153.294
Pfile5	2	4	4	0.16	24	37.747 3	0.16	25	37.747 3
Pfile6	2	5	4	0.17	20	51.782 6	0.16	26	51.782 6
Pfile7	2	6	4	0.18	23	93.009	0.17	20	142.179
Pfile8	3	6	5	0.41	31	165.985	0.40	32	160.639
Pfile9	3	7	5	0.47	55	120.028	0.46	39	119.82
Pfile10	3	8	5	0.51	40	167.868	0.50	50	181.68
Pfile11	3	7	6	0.71	24	155.308	0.71	28	155.308
Pfile12	3	8	6	0.83	42	139.006	0.86	59	126.007
Pfile13	3	10	6	0.99	50	89.904 7	1.08	78	90.28
Pfile14	5	10	10	13.55	89	344.858	13.03	69	375.056
Pfile15	5	15	12	31.95	115	403.565	32.18	94	407.887
Pfile16	5	15	14	53.78	185	409.71	55.03	156	394.27
Pfile17	5	20	16	144.40	359	285.434	102.46	180	294.623
Pfile18	5	20	18	127.79	162	170.348	137.20	178	154.548
Pfile19	5	25	20	239.24	192	374.05	236.5	214	373.75
Pfile20	2	25	22	311.22	296	651.716	307.67	255	725.695

3.2 分 析

在描述能力对比方面,DCIPS 可以用对象约束来描述状态对象之间的关系,这些对象约束还组成了不同的对象约束层次,使得对问题状态的描述可以在不同的层面上进行,这明显比通用规划系统 MIPS 要好.

在运行效率对比方面,在 Depots 问题上,DCIPS 效率明显比通用规划系统 MIPS 要高,扩展的节点数比 MIPS 要小,所求得的最优解值比 MIPS 也要好;在 DriverLog 问题上,大多数情况下 DCIPS 所求得的最优解比 MIPS 要好,因为是采用最优扩展法来对问题进行求解,所以 DCIPS 相应地要增加运行时间和扩展节点数;在 ZenoTravel 问题上,因为问题本身相对简单些,领域知识在其中所起到的作用不是太大,所以 DCIPS 和 MIPS 相差不大,但是 DCIPS 在最优解值方面还是比 MIPS 略胜一筹.

综上所述,在具体的问题领域求解中,DCIPS 效率比通用规划系统 MIPS 要高,这种效率的优势在复杂情况下越加显得明显.DCIPS 采用对象约束来表达状态中对象之间的关系;采用过程约束来表达动作之间的关系;采用时序约束表达动作与状态中对象之间的关系.在提高效率时,要采用到上述 3 种领域约束的作用,相应地增加了时间、空间代价.但一般具体的问题领域是属于比较复杂的情况,这种代价比规划回溯求解的时间代价要小得多,所以加入了领域约束的 DCIPS 运行效率就比 MIPS 要高.DCIPS 通过减少规划系统搜索的节点和增加启发信息来提高规划系统的效率,这些都使得 DCIPS 有所提高.

对于某些个别例子,DCIPS 的效率比 MIPS 略低,但是在大多数情况下,DCIPS 求得的最优解都比 MIPS 要小.这是因为,AIPS 2002 上的例子都是随机生成的,而 DCIPS 和 MIPS 都是采用启发信息来进行最优搜索,DCIPS 对 MIPS 的启发函数进行了改进,但是这些改进在规划搜索刚开始节点少的时候所起到的作用不大.因此,在实际问题搜索中,开始的时候根据启发信息选择的分支不一致,从而导致了最终结果的差异.

4 结论和相关工作

本文的贡献在于效率和方便添加领域知识这两个方面.在效率上,我们增加的仅仅是简单的近于常识的知识,对领域的依赖性很小,而且增加知识的量也很小,因此,系统应具有通用性,如果把具体领域专家的知识加进去,让系统进一步结合某类具体问题,效率还会进一步提高.在方便添加领域知识方面,经过我们的分析和整理,领域知识的表示与通用规划系统的描述语言十分类似,在通用规划系统中加上领域知识是非常方便的.比如,在上面 3 个测试用例中,我们只简单地采用了 7 条近于常识的领域知识,就使得规划系统的效率有所提高.

在领域依赖规划研究方面,与我们的研究工作相关的规划系统有 TLPLAN[16].它是基于每个问题的目标进行分析,主要是对目标进行时序扩展,人为给定目标完成的先后次序,从而提高规划系统的效率.DCIPS 与 TLPLAN 的区别在于,DCIPS 主要是对某个领域进行分析,不必针对特定的某个问题;而 TLPLAN 是针对特定的每个问题的目标集合.这样,如果在问题复杂的情况下,它的目标集合难以人为地分析其完成的先后次序,那么估计 TLPLAN 通过目标的时序扩展的方法可能起到的作用就不够明显.

如何使用领域知识,从而提高规划系统的描述能力和求解效率,这是一个包含范围很广且有相当难度的问题.目前,我们的研究工作只是勾画出轮廓,在不少方面仍未有深入的研究,与实用化仍有一定的距离.在未来的研究工作中,我们将在以下几方面着手深入研究,完善基于模型检测的 DCIPS 的功能,使其成为一个自动化的、功能强大的实用应用规划系统:(1) 统一的领域约束形式化表示.目前我们为了更为贴近于实际应用,对对象约束、过程约束和时序约束都采用了不同的形式化描述,但是三者是否存在同一的形式化描述?这是值得考虑的一项研究工作.(2) 领域约束的自动抽取.实现的关键技术是机器学习,即通过加入学习功能,使得 DCIPS 能从以前规划的结果中自动抽取出相关领域知识,并将其作用在规划求解上,从而提高效率.(3) DCIPS 商业化.本文研究的对象只是一个简化了的商业领域,完整的商业领域仍有很多重要的组成部分.在今后的研究工作中,继续向具体的商业应用方向发展.

References:

[1] Helmert M. Complexity results for standard benchmark domains in planning. Artificial Intelligence, 2003,143(2):219~262.

[2] Blum AL, Furst ML. Fast planning through planning graph analysis. Artificial Intelligence, 1997,90:281~300.

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

[4] Edelkamp S, Helmert M. The model checking integrated planning system. AI Magazine, 2001,22(3):67~71.

[5] Bertoli P, Cimatti A, Pistore M, Roveri M, Traverso P. MBP: A model based planner. In: Proc. of IJCAI 2001 Workshop on Planning under Uncertainty and Incomplete Information. Seattle, 2001.

[6] Jensen RM, Veloso MM, Bowling MH. OBDD-Based optimistic and strong cyclic adversarial planning. In: Proc. of the 6th European Conf. on Planning (ECP 2001). Springer-Verlag, 2001.

[7] Bryant RE. Graph-Based algorithms for boolwan function manipulation. IEEE Trans. on Computers, 1986,35(8):677~691.

[8] Fox M, Long D. PDDL2.1: An extension to PDDL for expressing temporal planning domains. 2002. http://www.dur.ac.uk/d.p.long/pddl2.ps.gz

[9] Haslum P, Scholz U. Domain knowledge in planning: Representation and use. In: Proc. of the ICAPS 2003 Workshop on PDDL. 2003.

[10] AIPS 2002 Competition Domains. 2002. http://www.dur.ac.uk/d.p.long/competition.html

[11] Garagnani M. Extending graphplan to domain axiom planning. In: Proc. of the 19th Workshop of the UK Planning and Scheduling SIG (PLANSIG 2000). Milton Keynes, 2000. 275~276.

[12] Garagnani M. A correct algorithm for efficient planning with preprocessed domain axioms. In: Bramer M, Preece A, Coenen F, eds. Research and Development in Intelligent Systems XVII (Proc. of ES 2000) Springer-Verlag, 2000. 363~374.

[13] Console L, Picardi C, Ribaudo M. Process algebras for systems diagnosis. Artificial Intelligence, 2002,142:19~51.

[14] Son TC, Baral C, McIlraith S. Planning with different forms of domain-dependent control knowledge—An answer set programming. In: Proc. of the 6th Int'l Conf. on Logic Programming and Nonmonotonic Reasoning, LPNMR 2001. Vienna, 2001. 226~239.

[15] Emerson EA. Temporal and modal logic. In van Leeuwen J. Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics. Amsterdam: Elsevier/MIT Press, 1990. 997~1072.

[16] Bacchus F, Kabanza F. Using temporal logics to express search control knowledge for planning. Artificial Intelligence, 2000,116: 123~191.