ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.9, September 2008, pp.2243−2264
DOI: 10.3724/SP.J.1001.2008.02243

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

智能规划器 StepByStep 的研究和开发*

吴向军[1], 姜云飞[2], 凌应标[3+]

[1](中山大学 软件学院,广东 广州 510275)

[2](中山大学 信息科学与技术学院 软件研究所,广东 广州 510275)

[3](中山大学 信息科学与技术学院 计算机科学系,广东 广州 510275)

Research and Development of StepByStep AI Planner

WU Xiang-Jun[1], JIANG Yun-Fei[2], LING Ying-Biao[3+]

[1](Software School, SUN YAT-SEN University, Guangzhou 510275, China)

[2](Software Research Institute, School of Information Science and Technology, SUN YAT-SEN University, Guangzhou 510275, China)

[3](Department of Computer Science, School of Information Science and Technology, SUN YAT-SEN University, Guangzhou 510275, China)

+ Corresponding author: E-mail: isslyb@mail.sysu.edu.cn

Wu XJ, Jiang YF, Ling YB. Research and development of StepByStep AI planner. *Journal of Software*, 2008, 19(9):2243−2264. http://www.jos.org.cn/1000-9825/19/2243.htm

Abstract: AI planner is one of the important representations of AI planning study, the performances of planner, the efficiency and quality of plan, represent the researches of AI planning directly. This paper introduces the architectures of AI planner and StepByStep planner in brief, then describes the methods and strategies adopted by SteByStep in detail, defines the knowledge tree of predicate for extracting domain knowledge. Based on knowledge tree of predicate, the planning tree of predicate is defined and some strategies are applied for constructing the planning tree fast. StepByStep adopts some planning strategies according to the planning trees. Finally, some experiments are taken for eight planners with three representative benchmark domains and their problems, the performances of StepByStep, the efficiency and quality of plan, are analyzed in detail. Experiments show that the strategies of StepByStep controls the process of planning the problems of the three domains well, and validate the guide effect of the domain knowledge in planning process.

Key words: artificial intelligence; AI planning; AI planner; planning domain; knowledge tree; planning tree; planning strategy

摘 要: 智能规划器是智能规划研究成果的重要表现形式,规划器的求解效率和规划质量是智能规划理论研究的直接反映.首先介绍智能规划器的一般结构和StepByStep规划器的总体结构,然后详细阐述StepByStep规划器各组成部分所采用的方法和策略,定义谓词知识树来提取领域知识.在谓词知识树的基础上定义谓词规划树,并用

* Supported by the National Natural Science Foundation of China under Grant No.60773201 (国家自然科学基金); the Special Foundation of 985 Project of SUN YAT-SEN University of China (中山大学 985 工程专项资金)

Received 2007-04-15; Accepted 2007-10-09

各种策略来提高规划树的生成效率.在谓词规划树的基础上设计 StepByStep 的规划策略,最后用 8 个规划器对 3 个具有代表性的基准规划领域及其规划问题进行实际的求解实验,分析了 StepByStep 规划器在求解效率和规划质量上的具体表现.实验数据表明,StepByStep 规划器的规划策略对 3 个不同规划领域都具有很好的指导作用,验证了领域知识在规划求解过程中的实际价值.

关键词: 人工智能;智能规划;智能规划器;规划领域;知识树;规划树;规划策略

中图法分类号: TP18 **文献标识码:** A

智能规划器是智能规划研究成果的重要表现形式,规划器的求解效率和规划质量直接反映了规划理论研究的研究成果.通过广大研究者的不懈努力,到目前已设计出数十种不同类型的智能规划器[1,2],并有相应的智能规划大赛 IPC(international planning competition)[3].智能规划不仅是具有理论研究价值的课题,而且是一个应用性较强的课题.随着规划系统效率的不断提高,该领域的研究成果有了一些重要的具体应用,尤其是与具体应用领域相关的规划系统,如美国国防部高级研究计划局主持的后勤支援计划——DART 系统,该系统能对多达 5 万台军事设备及其相关人员进行自动规划与调度,在数小时内完成传统方法需数周才能完成的任务.美国国家航天和宇宙航行局(NASA)在“太空一号”的控制器 RAX/PS 中采用了智能规划技术[4],欧洲 PLANET 项目也利用智能规划技术成功地实现“地球-1”探测器的自主控制飞行.

智能规划的研究方法有推理方法和程序化方法.推理方法的主要思想是用谓词公式来描述规划的初始状态、目标状态和规划动作,它把规划问题转换成逻辑推理问题,规划的动作序列就变成推理步骤.第一个采用推理指导规划过程的系统是问题解答系统 QA3[5].虽然该方法具有一定的理论意义,但其求解效率非常低下.

程序化方法是 Nilssion 等人在问题求解系统 STRIPS(stanford research institute problem solver)中所使用的求解方法[6].该方法是智能规划研究中的一个十分重要的研究成果,它用原子谓词集合描述状态,把动作描述成前提条件、增加表和删除表等 3 个部分.在当前规划状态下,若满足某动作的前提条件,则可执行该动作来修改状态,并得到执行动作后的规划状态.STRIPS 的这种动作描述方法和求解策略对此后的规划系统产生了深远的影响.另外,在此基础上也形成了一种描述规划领域和规划问题的描述语言——STRIPS 描述语言.

在智能规划研究中,研究者都知道领域知识在规划求解过程中的重要作用,并根据不同的研究策略采用不同的方式来体现领域知识的价值.规划器 STAN 第 1 次采用一个独立的领域分析工具 TIM[7]来分析领域定义和问题实例中的隐含信息,在规划中直接使用这些隐含信息来提高规划器的性能.图规划(GraphPlan)[8]是近年来智能规划研究中的一个重要方法,它利用动作和谓词之间的二类互斥关系来表达隐含的领域知识.这些互斥关系虽由动作描述和逻辑规则来获得,但它们都是与具体的规划实例相关的,不能反映规划领域中的一般规律.

在表达领域知识方面,有在领域描述中直接添加领域知识的,如:PDDL 描述语言用 Axiom 子句[9]来表达领域知识;有用时态模态逻辑规则来表示领域知识的,如:TALplaner[10,11]等;有把领域知识隐含在启发式函数中,并通过不同的加权来体现不同领域知识的作用,如:Blockbox[12,13],AltAlt[14,15]等,也有用二元判定图 BDD(binary decision diagram)[16]来反映领域知识的,如:MIPS[17,18]等.

对 STRIPS 类型的规划领域,我们采用谓词的相似关系来确定可实现同一谓词的所有规划动作,然后利用这些动作的公共前提和公共效果来提取规划领域的基本知识,用类似蕴涵公式的形式来表示领域知识[19],再利用这些被提取出来的基本知识和动作之间的相互作用进行逻辑推导以获得规划领域的复合知识[20].本文的规划器 StepByStep 将采用这种方式来表达和运用领域知识.

本文第 1 节简单介绍规划器的一般结构和 StepByStep 规划器的总体结构,说明规划器的输入/输出形式,并按“与规划领域的相关性”对规划器的种类进行分类.第 2 节定义谓词知识树,利用谓词知识树来提取领域知识和划分谓词的种类,说明各类谓词在规划求解过程中的作用.第 3 节在谓词知识树的基础上定义在当前状态下谓词的规划树,并用各种策略加速生成谓词规划树.第 4 节给出 StepByStep 规划器的规划策略,突出 StepByStep 是“以谓词为规划主体”的规划策略.最后用 7 个规划器与 StepByStep 规划器在相同环境下对 3 个具有代表性的基准规划领域及其规划问题进行实际的求解实验,根据各规划领域的特点详细分析了 StepByStep 的具体表现.

1 智能规划器的结构

1.1 智能规划器的一般结构

假设有一个规划领域,给定其一个规划问题的“初始状态”和“目标状态”.智能规划研究的问题是:在尽可能短的时间内求出满足该规划问题要求的一个动作序列,通过该动作序列能把“初始状态”转变成“目标状态”.智能规划的研究核心是规划策略的研究,其研究成果的有效性可由规划器的求解效率和规划质量来体现.智能规划器的一般结构如图 1 所示.

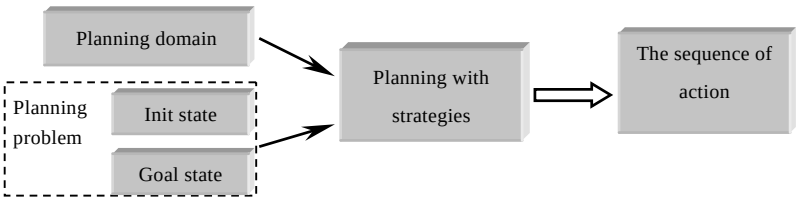

Fig.1 The general architecture of AI planner

图 1 智能规划器的一般结构示意图

由图 1 可知:规划领域和规划问题的描述是规划器的二组输入,规划的动作序列是其输出.

下面针对智能规划器各组成部分的内容和作用进行简单说明.

(1) 规划领域的描述

规划领域的描述部分主要包括:规划领域名、领域类型、领域谓词和领域动作等,其中,领域动作是规划求解的主体,也是规划器研究的主要对象.为了更好地描述规划领域的领域知识,规划领域的描述部分也在不断地扩充.比如,在规划领域中增加条件效果(conditional effect)、领域公理(axiom)、持续时间(duration)、数值表达式(numeric expression)等说明,这些说明信息增强了描述能力,也为规划求解带来了一些有用的领域信息.

规划领域的描述语言有:STRIPS,ADL(action description language)和 PDDL(planning domain definition language)等.STRIPS 是问题求解系统 STRIPS 中采用的描述语言[6],ADL 是在 STRIPS 基础上进行的扩展,它允许使用全称量词(universally quantified)、存在量词(existential quantified)和条件效果(conditional effect)等.PDDL 描述语言的表达能力覆盖了 STRIPS 和 ADL 的描述能力[9,21],它目前是智能规划大赛 IPC 所采用的规划领域描述语言.本文的规划器 StepByStep 也是以 PDDL 描述的规划领域来开展相应的研究.

(2) 规划问题的描述

规划问题的描述是对求解问题的描述,它主要包括:规划问题的初始状态、目标状态和所属的规划领域等.

一般情况下,任何一个规划领域,我们都可以设计出多个具体的规划问题,不同的规划问题需要用不同的规划动作序列来解决.附录 A 是 BlocksWorld 规划领域中的一个著名的规划奇异问题.

(3) 规划的求解策略

规划策略是智能规划研究的核心内容,规划策略的优劣将直接影响规划器的效率和规划的质量.

在规划策略方面,目前主要有:启发式状态搜索策略(HSP,GRT)、图规划策略(GraphPlan,STAN)、宽度优先状态空间搜索(PropPlan)、基于 SAT 求解技术(SATPlan)、线性时态逻辑(TALPlanner)、规划后处理策略(System R)等.在一些规划器中,其规划策略可能是多种求解策略的综合体,如:规划重写规则和局部搜索技术(PbR)、运用规划图和启发式状态空间搜索策略(AltAlt)、在规划图中运用局部搜索策略(LPG)[22]、基于模型检测和二分判定图 BDD 的求解方法(MIPS)等.

1.2 智能规划器的分类

智能规划的研究者一般会根据其研究思想开发出一个智能规划器来检验其研究成果的有效性,并在此基础上通过进一步的分析研究来完善其设计理念.随着各种规划策略的不断提出,相应的规划器也在不断地公布,

到目前为止,已公布了几十个不同类型的规划器[1,2].规划器的种类会因分类标准不同而不同,本文以规划器与规划领域的相关性为标准来进行分类.在这种分类标准下,规划器可分为两大类:与领域相关的规划器(domain-dependent planner)和与领域无关的规划器(domain-independent planner).

(1) 与领域相关的规划器

与领域相关的规划器在规划求解过程中人为地加入了领域知识、规则或常识,从而避免一些不必要的动作或推理,以达到提高规划求解效率的目的.

由于领域知识是多种多样的,不可能用一种方式来加入不同的领域知识.目前,加入领域知识的主要方式有:加入动作的时序,增加选择动作的先后顺序,加入新的谓词等.比如:为了说明在求解一个问题之前必须完成哪些任务,可在领域的定义中加入新谓词(SolvingGoal T_i Pd_u *Action*/*Null* Pd_v),其含义是:在规划状态 T_i 下,若需要实现目标 Pd_u,则必须先用动作 *Action*/*Null* 实现目标 Pd_v,其中,动作 *Null* 表示需用一组动作来实现.

例如,(SolvingGoal {(on *x y*)}$\cup T_j$(clear *y*) *Null* (clear *x*))的含义是:在含有谓词(on *x y*)的规划状态下,若要实现(clear *y*),则必须先用一组动作序列实现(clear *x*),然后才能实现(clear *y*).

由此可见:谓词 SolvingGoal 给出了问题求解的前后顺序,这显然为加速求解过程带来了有用的信息.

由于在规划器设计时加入了特定的领域知识,与领域相关的规划器一般都具有较高的规划效率,但这也限制了这类规划器的通用性.

(2) 与领域无关的规划器

与领域无关的规划器通常是采用与领域无关的规划策略(如:启发式搜索函数、图规划等)来指导问题求解.一般情况下,同一个(组)启发式函数很难对所有规划领域都能产生出高效的指导作用.因此,这类规划器的效率和规划质量会因规划领域的不同而有所差异.

对一些具体的应用,运用特定的领域知识设计出高效的规划器是系统开发者的设计目标.对智能规划的研究来说,开发出高效的、与领域无关的通用规划器是研究者所追求的研究目标.

1.3 规划器StepByStep的总体结构

在智能规划研究中,我们以与领域无关的规划器为设计目标.由于 STRIPS 类型的规划领域是研究智能规划的基础,所以,在设计规划器的处理能力方面,我们先以处理用 PDDL 语言描述的 STRIPS 规划领域为目标,在其之后再扩充到能够处理 PDDL 描述的所有规划领域.规划器 StepByStep 的总体结构如图 2 所示.

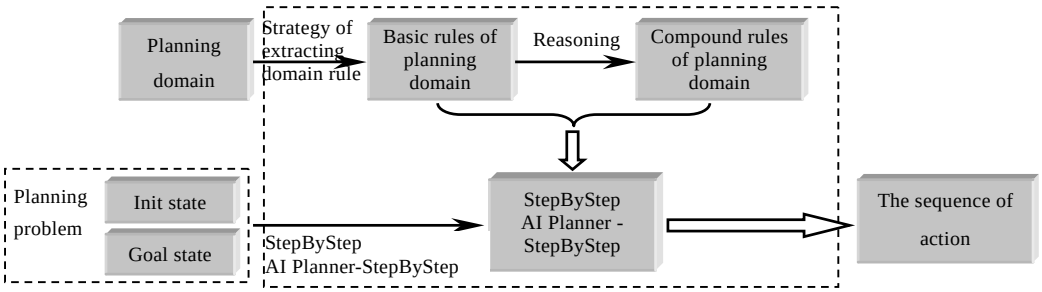

Fig.2 Architecture of StepByStep planner

图 2 规划器 StepByStep 的结构示意图

在规划器的设计中,我们先对规划领域中的动作定义进行处理,提取出可反映规划领域内在规律的基本规则.由于这些基本规则是直接从动作定义中获取的,它把隐含在动作定义中的领域规则显式地表现出来.在这些基本规则的基础上,再运用规划动作之间的相互作用进行适当的逻辑推理,从而获得反映规划领域更深层次的领域规则.在规划问题的求解部分,我们将充分利用自动提取的领域规则来指导规划求解.

一个规划领域一般会有多个具体的规划求解问题,同一个领域的不同规划问题,规划器 StepByStep 会用同一组领域规则来指导规划求解.由于领域规则是从领域定义中产生出来的,而非人为添加的,所以,不同规划领

域的规划问题,规划器 StepByStep 会自动采用不同的领域规则来指导规划求解.这种指导规划求解的领域知识具有可靠的领域背景,它仅与规划领域的定义信息有关,而与任何人为因素都无关.

2 规划领域的知识树

在一个规划领域中,一个动作的效果部分一般会含有多个领域谓词,一个领域谓词也会包含在多个动作的效果之中.因此,实现谓词的动作和领域谓词之间存在着“多对多”的关系.这种“多对多”关系增加了规划求解的难度,也为规划结果带来了多样性.在不同情况下选用不同的规划动作是智能规划研究的一种挑战.

对任意一个领域谓词,若要用动作来实现它,则需在规划领域动作中找出相应的领域动作.为避免在领域定义中反复进行这种类似的查找、匹配操作,我们用一种树形结构来存储可实现同一谓词的所有动作.

在本文的叙述中,我们用 $Para(Act)$,$PC(Act)$和 $E(Act)$分别表示规划动作 Act 的参数表、前提条件谓词集和动作效果的谓词集.

在文献[19]中,我们定义了两个谓词之间的相似关系“<~”,并证明了:若 $Pd_1<\sim Pd_2$,则一定存在置换σ,使得 $Pd_1=\sigma Pd_2$.

由此可知:若 $Pd_1<\sim Pd_2$,则谓词 Pd_1 是谓词 Pd_2 的一个实例(或变形).

定义 2.1. 假设:Pd 是一个谓词,S 是一个谓词集合,Act 是一个领域动作,σ是一个置换,

(1) σPd 表示谓词 Pd 的参数表用置换σ进行变换后的谓词;

(2) σS 是谓词集 S 中每个谓词用置换σ进行变换后所组成的集合,即:$\sigma S=\{\sigma Pd_i|Pd_i\in S\}$;

(3) σAct 是用置换σ对动作 Act 的参数表、前提条件谓词和动作效果谓词进行变换所得到的动作,即:

$$Para(\sigma Act)=\sigma Para(Act),PC(\sigma Act)=\sigma PC(Act),E(\sigma Act)=\sigma E(Act).$$

定义 2.2. 假设 Pd 和 Act 分别是一个规划领域的领域谓词和规划动作,若$\exists Pd_j\in E(Act)$,且 $Pd<\sim Pd_j$,则称规划动作 Act 可实现谓词 Pd.

所有可实现谓词 Pd 的动作集定义为 $Act(Pd)=\{Act_i \mid Act_i$ 是一个领域动作,且其可实现谓词 $Pd\}$.

定义 2.3. 假设 Pd 是一个规划领域的谓词,按照下面方法构造一个谓词-动作-谓词树 T:

(1) Pd 是树 T 根结点中的谓词;

(2) $\forall Act_i\in Act(Pd)$,存在置换σ_i,使得:$Pd\in E(\sigma_i Act_i)$,则动作$\sigma_i Act_i$ 是树 T 根结点的一个子结点,且 $PC(\sigma_i Act_i)$中的每个谓词都是动作$\sigma_i Act_i$ 的子结点;

称该谓词-动作-谓词树 T 为谓词 Pd 的知识树 KT(knowledge tree).

由定义 2.3 可知:知识树是一种只有两层结构的“And-Or”树.谓词知识树有两类结点:谓词结点和动作结点.动作结点表示实现其上层谓词的一个可选的规划动作,谓词结点是执行其下层动作所产生的一个效果,或执行其上层动作所需的前提条件.图 3 是 BlocksWorld 领域中谓词(holding x)的知识树**.

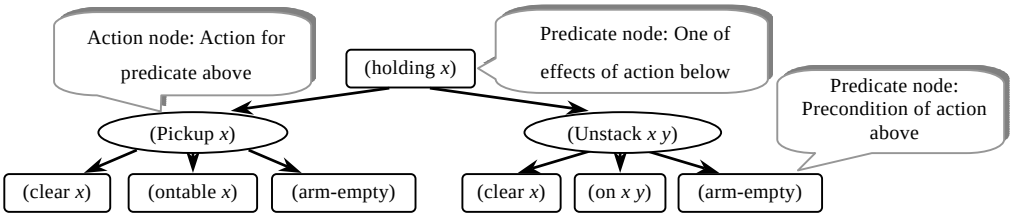

Fig.3 The knowledge tree of (holding x) in BlocksWorld

图 3 BlocksWorld 领域中谓词(holding x)的知识树

** 为了简洁表达知识树的主要思想,有关动作的效果部分没有在知识树上表达出来,但每个动作都会有其相应的动作效果.

由图 3 可知:在 BlocksWorld 领域中,可以实现谓词(holding *x*)的动作只有两个:(Pickup *x*)和(Unstuck *x y*).该知识树表达了实现谓词(holding *x*)的所有知识:实现该谓词的所有动作及其前提条件.在规划求解过程中,若需要实现谓词(holding *A*),则无需在领域动作表中再进行查找、匹配,可直接利用该知识树来确定实现目标的动作.这是把这样的谓词-动作-谓词树称作为“知识树”的一个重要原因.

利用谓词知识树还可获得实现该谓词的公共前提条件和公共效果谓词,从而为提取规划领域的基本知识提供了所需的领域信息.这是称该谓词-动作-谓词树为“知识树”的另一个重要原因.

由图 3 可以得到下面的信息:

(1) 实现谓词(holding *x*)的公共前提条件是{(clear *x*),(arm-empty)},公共效果是{(not (clear *x*)),(not (arm-empty))}.由这些信息可提取出“与谓词(holding *x*)相关”的直接伴随规则和直接阻碍规则[19,20].

(2) 动作(Pickup *x*)和(Unstack *x y*)仅有一个互不相同的前提条件,谓词(ontable *x*)和(on *x y*)分别是它们实现谓词(holding *x*)的特征前提.当(ontable *x*)成立时,只能选用动作(Pickup *x*);当(on *x y*)成立时,只能选用动作(Unstack *x y*).利用特征前提信息和各自不同的动作效果就可提取出“与谓词(holding *x*)相关”的条件伴随规则.

在一个规划领域中,不同谓词的知识树会有所不同.对任意一个领域谓词,不论其是正谓词,还是负谓词,都可构造其知识树.图 3 是正谓词(holding *x*)的知识树,它反映了实现该谓词的所有动作及其前提条件,图 4 是负谓词(not (holding *x*))的知识树,它反映了删除谓词(holding *x*)的所有动作及其前提条件.

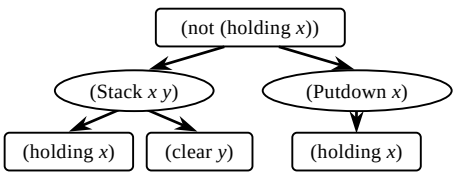

Fig.4 KT of (not (holding *x*)) in BlocksWorld

图 4 BlocksWorld 领域中(not (holding *x*))的知识树

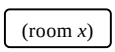

Fig.5 KT of (room *x*) in Gripper

图 5 Gripper 领域中(room *x*)的知识树

在有的规划领域中,存在谓词知识树仅有根结点的情况,这表明该谓词不会被任何领域动作所改变.在规划求解过程中,该谓词始终保持不变.图 5 是 Gripper 领域中谓词(room *x*)的知识树,它表示谓词(room *x*)不会被任何规划动作所增加.

分析 Gripper 领域的动作定义,我们还可以知道:负谓词(not (room *x*))的知识树也仅含根结点,即:谓词(room *x*)不会被任何规划动作所删除.因此,Gripper 领域中的谓词(room *x*)在任何规划求解过程中既不会被增加,也不会被删除.所以,该谓词的状态是静态不变的.

在其他规划领域中也会有类似的谓词,如:Monkey 规划领域中的谓词(location ?*x*),Hanoi 规划领域中的谓词(smaller ?*x* ?*y*),Logistics 规划领域中的谓词(truck ?*t*)和(airplane ?*p*)等.

定义 2.4. 假设谓词 *Pd* 是一个规划领域的谓词,

(1) 若谓词 *Pd* 和(not *Pd*)的知识树都仅含根结点,则称谓词 *Pd* 为静态谓词(static predicate),否则,称谓词 *Pd* 为动态谓词(dynamic predicate);

(2) 若谓词 *Pd* 的知识树仅含根结点,则称谓词 *Pd* 为非增谓词(non-added predicate);

(3) 若谓词(not *Pd*)的知识树仅含根结点,则称谓词 *Pd* 为非减谓词(non-deleted predicate).

有关定义 2.4 的含义说明如下:

(1) 若谓词 *Pd* 是非增谓词,则表示其在规划求解过程中肯定不会被增加到规划状态之中,但可能被删除,并且一旦被删除,此后就再也不会被添加到规划状态之中.所以,前提条件中有谓词 *Pd* 的动作一定要安排在删除谓词 *Pd* 的动作之前.

(2) 若谓词 *Pd* 是非减谓词,则表示其在规划求解过程中一定不会从规划状态中删除掉,但有可能被增加,并且一旦在规划状态中加入了该谓词,则该谓词就一直存在.所以,前提条件中有谓词(not *Pd*)的动作一定要安排在增加谓词 *Pd* 的动作之前.

谓词知识树反映的是实现树根谓词的所有知识,其根结点中的谓词通常含有未实例化的参数,但在规划求解时需要实现的谓词都是已实例化的.所以,在运用知识树时,还需要进行参数置换操作.

定义 2.5. 假设 KT 是谓词 *Pd* 的知识树,σ是一个置换,σ*KT* 是用置换σ对知识树 KT 中的所有动作都进行置换所得到的树,则σ*KT* 就是谓词σ*Pd* 的知识树.

图 3 是谓词(holding *x*)的知识树 KT.若要实现谓词(holding *A*),则需要一个实例化置换σ={*A*/*x*}.显然,σ*KT* 就是谓词(holding *A*)的知识树,如图 6 所示.

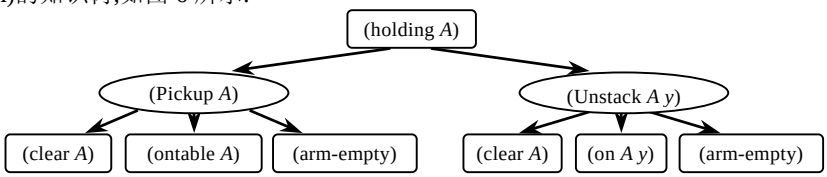

Fig.6 KT of (holding *A*) in BlocksWorld

图 6 BlocksWorld 领域中谓词(holding *A*)的知识树

对任何一个规划状态,本文用该状态所含的谓词集合来代表,且用符号 T_i 表示第 *i* 个规划状态,其中:T_0 表示规划问题的"初始状态(init state)",T_G 表示其"目标状态(goal state)".

例如,对附录 A 的规划求解问题,其"初始状态"和"目标状态"如下:

$$T_0=\{(\text{clear } C),(\text{on } C\ A),(\text{ontable } A),(\text{clear } B),(\text{ontable } B),(\text{arm-empty})\}$$

$$T_G=\{(\text{on } A\ B),(\text{on } B\ C)\}$$

下面利用谓词的知识树来研究在规划求解过程中,静态谓词所具有的特殊性质.

引理 2.1. 假设存在谓词 Pd_i 和静态谓词 *Pd*,若 $Pd_i<\sim Pd$,则谓词 Pd_i 一定不可用动作来实现.

证明:

由"$Pd_i<\sim Pd$"可知:一定存在一个置换σ,使得:$Pd_i=\sigma Pd$.

由"谓词 *Pd* 是静态谓词"和定义 2.4(1)可知:谓词 *Pd* 的知识树仅有根结点.

由定义 2.5 可知:谓词σ*Pd* 的知识树也仅有根结点,即:谓词σ*Pd* 不会由任何规划动作来实现.

所以,谓词 Pd_i 一定不可用规划动作来实现. □

定理 2.1. 假设谓词 *Pd* 是静态谓词,若$\exists Pd_i\in T_G-T_0$,且 $Pd_i<\sim Pd$,则目标状态 T_G 一定是不可实现的.

证明:

由已知条件"$Pd_i\in T_G-T_0$"可知,$Pd_i\in T_G$,$Pd_i\notin T_0$.

所以,在规划求解过程中,需要用规划动作来实现谓词 Pd_i.

由引理 2.1 可知:谓词 Pd_i 不可用任何规划动作来实现.

所以,目标状态 T_G 一定是不可实现的. □

例如,在 Gripper 领域的任何一个规划问题中,若(room *A*)$\in T_G-T_0$,则由定理 2.1 可知,该目标状态 T_G 一定是不可实现的.

引理 2.1 说明,静态谓词的任何实例(或变形)都是不可实现的,定理 2.1 说明,需实现静态谓词实例的目标状态一定是不可实现的.

3 规划树的设计思想

规划器 StepByStep 的求解过程是在规划树上进行的,规划树是在当前规划状态下利用谓词知识树产生出来的.在生成规划树时,我们采用各种策略来提高它的生成效率.本节介绍子目标规划树及其生成策略.

3.1 子目标的规划树

在下面的叙述中,称目标状态 T_G 中的每个谓词为子目标.若目标状态中的每个子目标都实现了,则该目标状态也就实现了.因此,我们把整个规划问题的求解分解成每个子目标的求解.

假设谓词 Pd 是一个领域谓词,T_i 是一个规划状态,若 $Pd \in T_i$,则称在状态 T_i 下,谓词 Pd 为“真”,(not Pd)为“假”,否则,在状态 T_i 下,称谓词 Pd 为“假”,谓词(not Pd)为“真”.

由此可见:谓词的“真”/“假”与具体的规划状态有关.在不强调具体规划状态时,本文把“在状态 T_i 下,谓词 Pd 为‘真’/‘假’”简写为“谓词 Pd 为‘真’/‘假’”.

定义 3.1. 假设谓词 Pd 是规划领域的一个谓词,当前规划状态为 T_i,按照下面方法递归地构造一个谓词-动作-谓词树 PT:

(1) PT 的根结点是含有谓词 Pd 的谓词结点;

(2) 若 $Pd \notin T_i$,则 PT 的第 1 层是谓词 Pd 的知识树,对谓词 Pd 知识树的每个叶结点谓词 Pd_j,递归地构造谓词-动作-谓词树 PT_j;

称该谓词-动作-谓词树 PT 为谓词 Pd 在状态 T_i 下的规划树 PT(planning tree).

由定义 3.1 可知,谓词 Pd 的规划树是由若干个谓词知识树组成的.规划树按“层”来组织,每层都是相应谓词的知识树.在谓词规划树的生成过程中,将利用各种因素对谓词知识树进行剪枝(见第 3.2 节).所以,每层谓词知识树可能是原知识树的一部分.谓词 Pd 的规划树形式如图 7 所示.

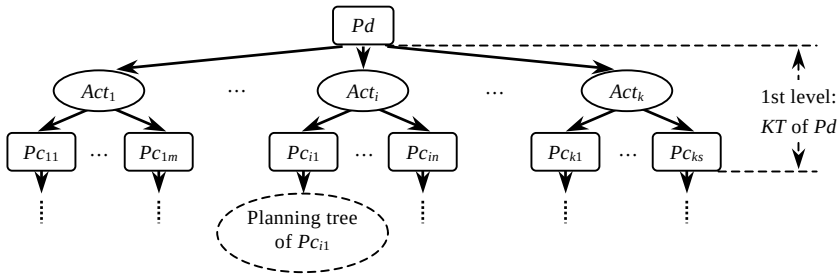

Fig.7 The planning tree of Pd

图 7 谓词 Pd 的规划树示意图

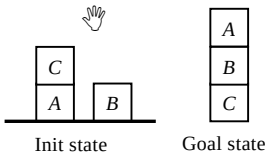

Fig.8 States of Sussman anomaly

图 8 Sussman 问题的状态示意图

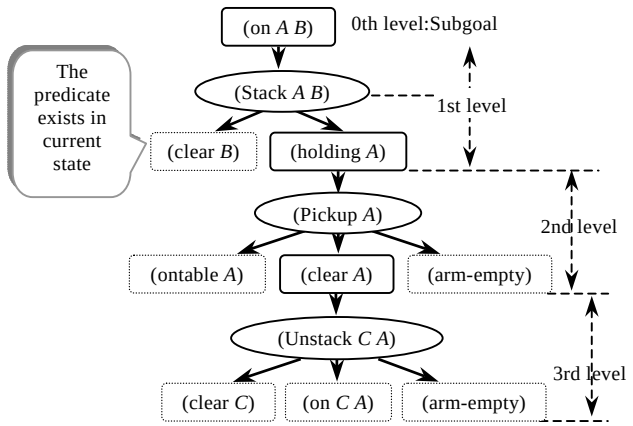

Fig.9 Planning tree of subgoal (*on A B*) under initial state

图 9 子目标(*on A B*)在初始状态下的规划树

由图 7 可知,规划树的结构与“And-Or”树相一致.实现谓词 Pd 的动作 $Act_1,Act_2,\ldots,Act_k$ 之间是“或”关系,其含义是:只要执行其中一个动作就可以实现谓词 Pd.同一个动作 Act_i 的前提条件 $Pc_{i1},Pc_{i2},\ldots,Pc_{in}$ 之间是“与”关系,其含义是:只有动作 Act_i 的所有前提条件在当前规划状态下都存在,该规划动作才能被执行.

在生成谓词规划树时,把谓词知识树当作一个整体进行扩展.树中谓词结点内的谓词是其低层动作的一个效果,同时又是其上层动作的一个前提条件(根结点谓词除外).

图 8 是 BlocksWorld 领域 Sussman 问题的初始状态和目标状态的示意图.在该初始状态下,可根据定义 3.1 构造出其子目标(on A B)的规划树,如图 9 所示.图 9 中的第 1 层是谓词(on x y)的实例化知识树,第 2 层是谓词(holding x)实例化知识树的一部分,其他部分被规划树的生成策略剪枝掉(如图 14 所示),第 3 层是谓词(clear x)实例化知识树的一部分,其他部分也被生成策略剪枝掉.

定义 3.2. 下面递归定义规划树 PT 中各类结点的高度 H:

(1) 若谓词 Pd 是规划树中的叶结点,则,$H(Pd)=0$;

(2) 若动作 Act_i 有 k 个前提条件为 $Pc_1,Pc_2,\ldots,Pc_k$,则,$H(Act_i)=\max(H(Pc_1),H(Pc_2),\ldots,H(Pc_k))$;

(3) 若实现谓词 Pd 有 m 个动作 $Act_1,Act_2,\ldots,Act_m$,则,$H(Pd)=\min(H(Act_1),H(Act_2),\ldots,H(Act_m))+1$.

由定义 3.2 可知:动作的高度是其所有前提条件谓词结点高度的最大值,谓词结点的高度是可实现该谓词的所有动作高度的最小值加 1,如图 10 所示.

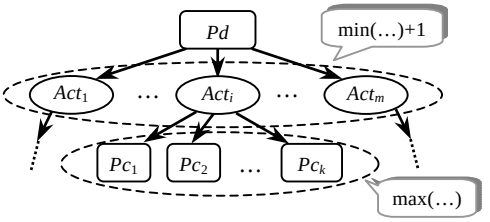

Fig.10 Height of node of planning tree

图 10 规划树中结点高度的示意图

谓词结点的高度反映了在当前规划状态下该谓词至少需要多少步动作才有可能被实现,越高的谓词也就越难被实现,所以,我们必须先选择高度低的谓词来实现,即先选择高度为 1 的谓词来实现.

定理 3.1. 假设谓词 Pd 是规划树谓词结点中的谓词,若 $H(Pd)=1$,则在当前规划状态下,谓词 Pd 一定可用一个动作来实现.

证明:

由定义 3.2(3)可知,在所有可实现谓词 Pd 的动作中,至少存在一个动作 Act_i,使得 $H(Act_i)=0$.

由定义 3.2(2)可知,$\forall Pc_j \in PC(Act_i)$,都有 $H(Pc_j)=0$.

由定义 3.2(1)可知,在当前规划状态下,动作 Act_i 的所有前提条件都已存在.

所以,在当前规划状态下,可执行动作 Act_i.执行该动作后,谓词 Pd 一定被实现. □

定理 3.1 说明:在当前规划状态下,谓词 Pd 一定可用一个动作来实现,但这并不表示实现后,谓词 Pd 就一直存在,它还可能被其他动作删除掉.这样的变化并不与“定理 3.1”的含义相矛盾.

利用定理 3.1,称高度为 1 的谓词结点为可实现的谓词结点,对应的谓词为可实现的谓词.

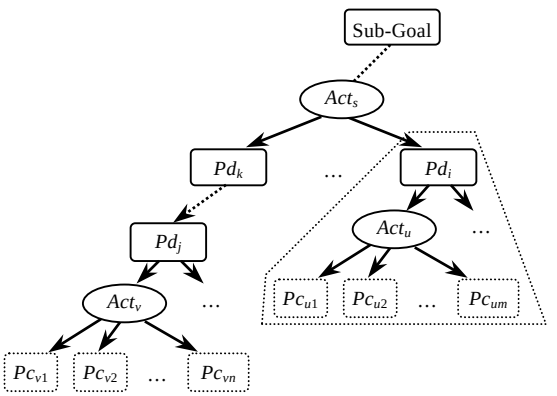

Fig.11 Two realizable predicates Pd_i and Pd_j

图 11 两个可实现谓词 Pd_i 和 Pd_j 的示意图

对任意一棵规划树,其所有高度为 1 的谓词在当前规划状态下都可用一个动作来实现,但这些能被实现的谓词并不表示它们能被“并行”实现,也不表示所有可被实现的谓词就一定都应该被实现.这是因为在采用一个规划动作实现了一个谓词的同时,该动作可能会破坏了实现其他谓词动作的前提条件.

假设:在规划树中有两个谓词 Pd_i 和 Pd_j,且 $H(Pd_i)=H(Pd_j)=1$,其中:结点 Pd_i 离根结点近,结点 Pd_j 离根结点远,如图 11 所示.

若先用动作实现谓词 Pd_i,那么,在实现谓词 Pd_k 的过程中,有可能会删去谓词 Pd_i,从而使前面实现 Pd_i 的动作变得多余,所以,需要从离根结点较远的谓词 Pd_j 向上逐步实现.

在规划树中,为了表示谓词结点离根结点的距离,下面从根结点依次向下定义谓词结点的层次.

定义 3.3. 假设:谓词 Pd 是规划树中一个谓词结点中的谓词,下面递归地定义谓词 Pd 的层次 L:

(1) 若谓词 Pd 是规划树根结点的谓词,则 $L(Pd)=0$;

(2) 若谓词 Pd 是规划树叶结点的谓词,则 $L(Pd)=-1$;

(3) 若谓词 Pd 是规划树分支结点的谓词,且动作 Act_i 是实现它的一个动作,该动作有 k 个前提条件 Pc_1, Pc_2,…,Pc_k,则 $L(Pc_j)=L(Pd)+1$,$j=1..k$.

定义 3.4. 称规划树 PT 中谓词结点层次的最大值为规划树的层次,记为 $L(PT)$.

定理 3.2. 假设有规划树 PT,且 $L(PT)\geq 0$,若存在 PT 中的谓词 Pd,使得 $L(Pd)=L(PT)$,则 $H(Pd)=1$.

证明:

假设:在规划树 PT 中存在谓词 Pd,使得 $L(Pd)=L(PT)$.

由已知条件 $L(PT)\geq 0$,$L(Pd)=L(PT)$和定义 3.3 可知,谓词 Pd 一定不是叶结点中的谓词.

由定义 3.2 可知,$H(Pd)>0$.

不妨再设 $H(Pd)\geq 2$.

由定义 3.2(3)可知:在所有可实现谓词 Pd 的动作中,至少存在一个动作 Act,使得 $H(Act)\geq 1$.

再由定义 3.2(2)可知,在动作 Act 的前提条件中,至少存在一个前提条件 Pc_j,使得 $H(Pc_j)\geq 1$.其中谓词 Pd,动作 Act 和前提条件 Pc_j 之间的关系如图 12 所示.

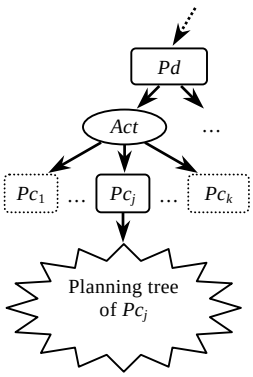

Fig.12 Relation among Pd, Act and Pc_j

图 12 谓词 Pd,Act 和 Pc_j 的关系示意图

所以,谓词 Pc_j 在当前状态下不可能为“真”,即谓词 Pc_j 所在的结点不可能是叶结点.

由定义 3.3(3)可知,$L(Pc_j)=L(Pd)+1=L(PT)+1$,即谓词 Pc_j 的层次更大,这显然与 $L(PT)$的定义相矛盾.

所以,$H(Pd)<2$.

因此,$0<H(Pd)<2$,即 $H(Pd)=1$. □

由定理 3.2 和定理 3.1 可知,离根结点最远的谓词结点一定是可实现谓词.

在规划树 PT 中,假设存在谓词 Pd_i 和 Pd_j,且 $H(Pd_i)=H(Pd_j)=1$,若 $L(Pd_i)>L(Pd_j)$,则在选择时,先实现谓词 Pd_i,除非谓词 Pd_j 是“非增谓词”或“非减谓词”.这是对所有可实现的谓词进行选择的策略之一.

对规划树,我们用各类结点的高度对其进行剪枝,用谓词的层次来对其所有可实现的谓词进行选择.

3.2 规划树的生成策略

子目标的规划树是规划器 StepByStep 进行规划求解的基础,它反映了在当前规划状态下实现该子目标可能采用的各种动作.子目标规划树的生成效率将直接影响规划求解的效率.因此,在子目标规划树的生成过程中,我们采用多种策略删除一些不必要的规划子树,以达到缩短规划树生成时间的目的.

(一) 优先处理静态谓词

由静态谓词的定义(定义 2.4)可知,静态谓词所表达的领域事实上在整个规划过程中是保持不变的.该类谓词参数表中未实例化的参数只能通过参数匹配来实例化,不可能通过动作来实现.若通过其他谓词的实现来确定静态谓词参数的取值,则可能会产生出无法实现的静态谓词,从而使前面的动作变成多余.

下面用 Logistics 领域中的一个问题来进行说明.

假设:当前需要把物体 P1 移到位置 Loc 处,即实现谓词(at P1 Loc).

实现谓词(at P1 Loc)的部分动作序列如下所示:

(at P1 Loc)

1) (Unload P1 ?*t* Loc)

PC: (in P1 ?*t*) (truck ?*t*) (at ?*t* Loc)

…

由定义 2.4 可知,谓词(truck ?*t*)是静态谓词,它表示“?*t*”是卡车(truck).

若按照动作 Unload 前提条件的定义次序来进行处理,则可能会先产生出实现谓词(in P1 Airplane)的规划树(σ={Airplane/?t},该谓词的含义是物体 P1 在飞机 Airplane 之中),然后再确定谓词(truck Airplane).显然,谓词(truck Airplane)是不成立的,也不可能用任何动作来实现.所以,生成谓词(in P1 Airplane)的规划树是多余的.

由上面的分析可知,在处理动作 Unload 的前提条件时,首先应该对静态谓词(truck ?t)进行参数匹配,比如:σ={Bos-Truck/?t},然后再对谓词(in P1 Bos-Truck)和(at Bos-Truck Loc)进行处理.若它们不在当前状态之中,则可用动作(或动作序列)来实现.

(二) 动作前提条件的排序

在研究规划领域时我们发现,领域设计者在编写动作时,若对动作的前提条件有意识地根据它们的重要性进行安排(如:重要的前提条件放在前面),则可利用这种有意的安排来提高规划效率.这种重要性安排也是设计者把其领域知识隐含在动作描述中的方法之一.

从表面上来看,动作前提条件之间是平行关系,不论先满足哪个前提,只要所有前提条件都满足了,该动作就可以被执行.但实际上,很多动作的前提条件之间都不是平行关系,而是有先后次序的.只有按照这种先后次序来满足动作的前提条件,所得到的动作序列才可能没有冗余操作.

下面用 Monkey 规划领域中一个简单直观的规划动作来说明,在其他规划领域中也存在类似的情况.

例如:在 Monkey 规划领域中,动作(grab-bananas ?y)的前提条件是(location ?y),(at bananas ?y),(onbox ?y)和(hasknife).

由静态谓词的特点和领域知识树可知,谓词(location ?y)和(at bananas ?y)只能用变量匹配来确定.

下面来分析谓词(onbox ?y)和(hasknife)的先后次序.

从动作定义的语义来看,在任意规划状态下,当谓词(onbox ?y)和(hasknife)都为假时,则需要用动作或动作序列把它们都转变为真,但在动作定义中没有明确这两个谓词被实现的先后次序(不能用书写顺序来代表,除非在 PDDL 描述语言中有特殊的语义说明).从基本的语义来看,这两个前提谓词是平行的,但实际上,它们之间隐含着先后次序.

由文献[20]可知,存在绝对阻碍规则:(onbox y)$_{\{(\text{not (onbox } y)),(\text{on-floor}),(\text{not (at Monkey } x)),(\text{not (hasknife)})\}}$=\\=>(hasknife),其规则条件是:猴子,刀和箱子在不同地方.

当谓词(hasknife)为假时,若猴子先爬上箱子,即先实现谓词(onbox ?y),那么,在后面的规划状态下一定无法使谓词(hasknife)为真.因为刀不在当前位置,猴子需要再下来到存放刀的位置,但领域定义中没有猴子从箱子上下来的动作描述.即使领域中定义了猴子从箱子上下来的动作,那么,猴子先前爬上箱子的动作也就是多余的,即先实现谓词(onbox ?y)是不必要的.

所以,动作(grab-bananas ?y)的前提条件次序应是(location ?y),(at bananas ?y),(hasknife)和(onbox ?y).

由此可见,在规划领域定义中,其规划动作的前提条件之间不能仅理解为平行关系,它们之间还可能存在串行关系,或存在二者兼有的关系,即一部分前提条件之间是平行的,另一部分前提条件之间是串行的.但在动作定义时,领域设计者无法用 PDDL 来描述动作前提条件之间的这种关系.我们在设计规划器 StepByStep 时,利用所提取的领域知识来明确动作前提条件之间的实现次序,这样就减少了一些不必要的规划动作.

(三) 循环谓词的剪枝策略

假设:在第 v 层,谓词 Pd_j 不在当前规划状态中,在谓词 Pd_j 的知识树中有动作 Act_k,且谓词 Pd 是该动作的一个前提条件.

若动作 Act_k 的前提条件 Pd 与第 s 层中的“祖

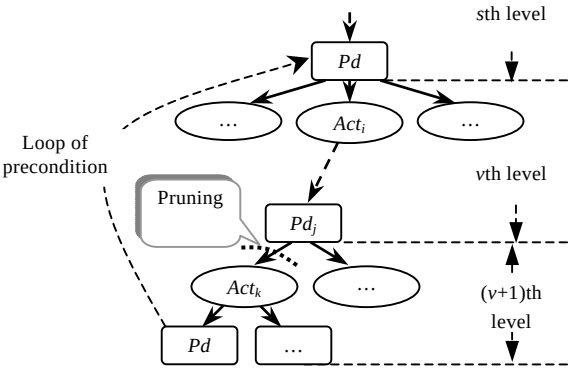

Fig.13 Loop between target and precondition

图 13 循环前提条件的示意图

先”谓词 Pd 相同,则形成一个谓词-动作-谓词的循环:$Pd–Act_i–…–Pd_j–Act_k–Pd$.其含义是:在第 s 层实现谓词 Pd,则需在其下面的第 v+1 层先实现谓词 Pd,这显然是一种循环前提条件.所以,谓词 Pd_j 不可能由执行动作 Act_k 来实现,即谓词结点 Pd_j 下面的动作 Act_k 应被删除.循环谓词的含义和剪枝策略如图 13 所示.

对附录 A 中的 Sussman 奇异问题,下面利用子目标(on *A B*)规划树的生成过程来说明循环谓词的剪枝策略.

在生成子目标(on *A B*)规划树时,第 1 层中谓词(clear *B*)为“真”,谓词(holding *A*)为“假”,所以,在第 1 层要用谓词(holding *A*)的知识树(如图 6 所示)来进行扩展,实现谓词(holding *A*)的动作有(Pickup *A*)和(Unstack *A y*).

执行动作(Pickup *A*)需要前提条件(ontable *A*),(arm-empty)和(clear *A*),如图 14 中的第 2 层所示.

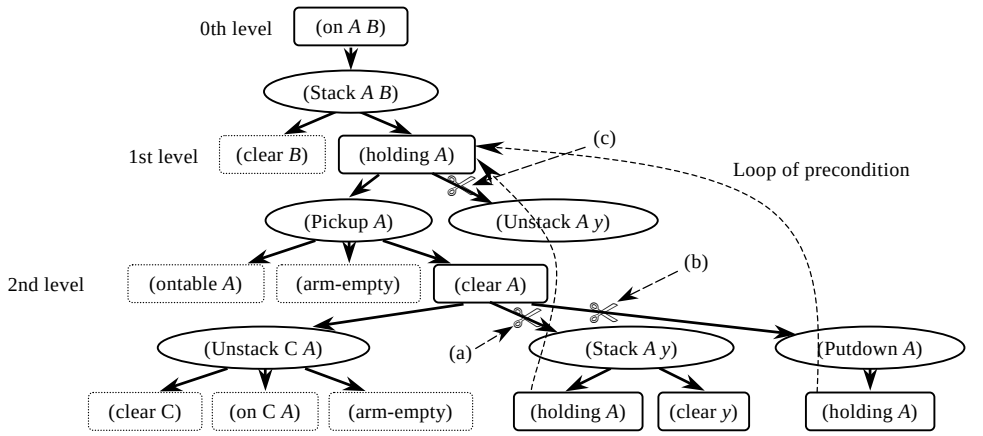

Fig.14 Loop predicate while building planning tree

图 14 生成规划树时存在循环谓词的示意图

第 2 层中的谓词(clear *A*)为假,根据 BlocksWorld 的领域定义和知识树的定义(定义 2.3)可知,在谓词(clear *x*)的知识树中有 3 个动作(Unstack *y x*),(Stack *x y*)和(Putdown *x*).对该知识树进行实例化置换{*A*/*x*}后可得图 14 中谓词(clear *A*)的知识树.

对于动作(Stack *A y*)和(Putdown *A*),它们又都需要前提条件(holding *A*).这就形成了:在规划树第 1 层需要实现谓词(holding *A*),则就要在第 2 层有谓词(holding *A*),显然,谓词-动作-谓词序列:(holding *A*)–(Pickup *A*)–(clear *A*)–(Stack *A y*)[(Putdown *A*)]–(holding *A*)就是一个谓词循环,所以,第 2 层下面的动作(Stack *A y*)和(Putdown *A*)都是应该被剪枝的,如图 14(a)和(b)所示.

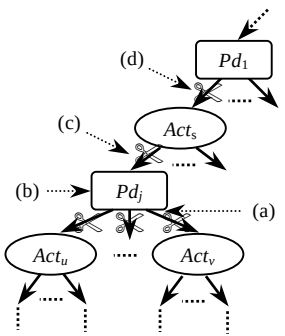

Fig.15 Chain-Reacting while deleting node of planning tree

图 15 删除动作的连锁反应示意图

在规划树的第 1 层,谓词(holding *A*)的知识树中还可用动作(Unstack *A y*)来实现谓词(holding *A*)(如图 6 所示),同样可利用循环谓词的剪枝策略剪去该动作分支.如图 14(c)所示.

在生成谓词规划树过程中,当删除某个动作时可能会引起一系列的连锁反应.下面利用图 15 对这种连锁反应作一般性的说明.

若谓词结点 Pd_j 下面的所有动作都被删除掉(图 15(a)),则在当前状态下,谓词 Pd_j 不可能用任何动作来实现,即谓词 Pd_j 不可能变为真(图 15(b)),所以,以谓词 Pd_j 为前提条件的动作 Act_s 一定不会被执行(图 15(c)),动作 Act_s 的其他前提无须再考虑就可直接删除动作 Act_s(图 15(d)).对动作 Act_s 再向上确定谓词 Pd_1 的其他实现情况,并根

据具体情况再进行相应的删除操作,直到不能进行删除为止.

(四) 剪除高的规划子树

假设在生成子目标 Pd_1 的规划树时,当前要展开的谓词为 Pd_i,实现谓词 Pd_i 的可选动作为 $Act_1,Act_2,\ldots,Act_k$.可选动作 Act_1 的规划子树为 PT_1.

由定义 3.2(3)可知,$H(Pd_i)\leq H(Act_1)+1$,谓词结点 Pd_i 规划树的高度至多为 $H(Act_1)+1$.

在生成实现谓词 Pd_i 的其他规划树时,可用高度 $H(Pd_i)$来进行限制,如图 16 所示.

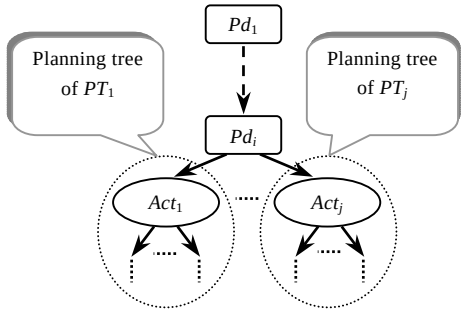

Fig.16 Some actions for Pd_i

图 16 可实现谓词 Pd_i 的动作示意图

任取实现谓词 Pd_i 的可选动作 Act_j,在对其生成规划树 PT_j 时,需考虑下列两种情况:

(1) 若 PT_j 不能在高度 $H(Pd_i)-1$ 内获得,则无须再继续生成,因为已有动作可在较少步数内实现谓词 Pd_i;

(2) 若 PT_j 能在 $H(Pd_i)-1$ 之内获得,即 $H(Act_j)<H(Pd_1)-1$,则 $H(Pd_i)=H(Act_j)+1$,因为有一个更矮的规划树来实现谓词 Pd_i.

在最好情况下,若 $H(Act_1)\leq min(H(Act_2),H(Act_3),\ldots,H(Act_k))$,即:第 1 个动作的规划树就是生成谓词 Pd_i 的最矮规划子树,实现谓词 Pd_i 的其他规划子树都被 $H(Pd_1)$剪枝掉,这就加快了谓词 Pd_i 规划树的生成过程.

在最坏情况下,若 $H(Act_1)\geq H(Act_2)\geq\ldots\geq H(Act_k)$,即动作规划树的高度是递降的.由于 $H(Act_i)\geq H(Act_{i+1})$,所以,动作 Act_{i+1} 的规划子树不会由前面的 $H(Act_i)$剪枝掉,动作 Act_{i+1} 的规划树会完整地生成出来,即所有动作前提条件的规划树都会被完整地生成出来.

为避免最坏情况出现,我们可采用宽度优先的生成策略来生成规划子树,并使用α-β剪枝策略来修剪所有不必要的分支,从而提高规划子树的生成效率.

4 规划器 StepByStep 的规划策略

智能规划研究的核心是规划策略,规划策略的优劣将直接影响规划器的求解效率和规划质量,研究人员会根据不同的研究理念来设计规划策略.

下面介绍规划器 StepByStep 的规划策略.通过对求解策略的描述,我们不难知道,规划器 StepByStep 的设计理念与现有规划器的设计理念有着本质的区别.

4.1 领域知识的自动提取策略

在智能规划研究中,我们知道:要想提高规划器的效率,规划领域知识的应用是不可缺少的.在其他规划器的设计中会采用各种方法来设计启发函数,并试图通过启发函数的计算来达到应用领域知识的目的.这种方法在规划过程中也取得了一定的成效,但同一个(或一组)启发函数很难对各类规划领域都能起到很好的作用.这些启发函数对某些规划领域可能有很好的启发作用,但对另一些规划领域未必能起到很好的作用.

在经典的图规划策略中,在用规划图进行规划求解时,通过定义动作和谓词、谓词和谓词之间的互斥关系来指导规划求解.这种互斥关系虽然在一定程度上反映了规划领域的内在规律,但它只对具体的规划图起作用.互斥关系是领域知识的一种具体表现,不能由这些具体的互斥关系抽象出规划领域的一般性领域知识.

为了利用领域知识,我们在设计规划器 StepByStep 时,首先利用领域知识提取策略从领域定义中自动提取领域规则,并用特定的形式来表达领域知识,然后利用规划动作之间的相互作用来进行逻辑推理,并得到反映规划领域的更深层次的领域知识.我们采用的领域知识提取策略是与规划领域无关的,其提取出来的领域知识是与规划领域相关的,对不同的规划领域会得到不同的领域知识.

对同一规划领域的不同规划问题进行求解时,规划器 StepByStep 会采用相同的领域知识来指导求解过程,但在求解不同规划领域的规划问题时,规划器会由不同的领域知识来指导求解.显然,这种用与规划领域相关的

领域知识来指导规划过程要比使用通用启发函数来指导规划求解具有更强的针对性.

4.2 目标状态的求解策略

在现有规划器的求解过程中,基本上把初始状态和目标状态看成一个整体.在利用规划动作对规划状态进行演变时,若当前规划状态包含了目标状态中的所有谓词,那么,目标状态就被实现了,状态演变过程中的动作序列就是实现目标的一个“规划”.

规划器 StepByStep 的求解是在子目标的规划树(第 3 节)上进行的,规划树的建立是以下列常识(common sense)为理论依据的.这 3 个常识是:

(1) 若目标状态是可实现的,则目标状态中的每个子目标都是可实现的;

(2) 若某个子目标是可实现的,则一定存在一棵高度有限的规划树来实现该子目标;

(3) 若某个子目标的规划树为空,则该子目标是已实现的,或其是不可实现的.

在建立一个子目标的规划树时,不考虑其他子目标的实现情况.这样,每个子目标规划树的状态空间都是有限的,所有子目标规划树的状态空间之间是“加”的关系,不存在多目标之间相互组合所产生的组合爆炸问题.

附录 B 是 BlocksWorld 领域 Sussman 奇异问题的子目标规划树.由定理 3.1 可知,在初始状态下,子目标(on *A B*)和(on *B C*)的规划树中谓词(clear *A*)和(holding *B*)都是可实现的.对可实现的谓词集{(clear *A*),(holding *B*)}进行选择来确定当前所要实现的谓词.若筛选后的候选谓词集中存在多个谓词,则这些谓词存在着同时被实现的可能性.

综上所述,我们的规划策略是:在当前规划状态下(规划求解开始时,当前规划状态为初始状态),用目标状态中的每个子目标建立规划树,然后把所有规划树中可实现的谓词组成一个候选谓词集合.利用领域知识排除一些不应“实现”的谓词,使候选谓词集中剩下的谓词都具有并行实现的可能性.

4.3 规划过程中的规划主体

在规划求解过程中,我们把从初始状态变为目标状态的每个中间步骤称为一个规划步,规划策略的不同将直接影响求解过程中的每个规划步.为了说明规划器 StepByStep 的规划策略与现有智能规划器的差异,我们以规划步的不同处理对象来划分规划方法:动作规划方法和谓词规划方法.

动作规划方法是指在规划步中以确定每步的规划动作为策略的规划方法,每个规划步是一个确定性的规划动作,该规划步称为动作规划步.

谓词规划方法是指在规划步中以确定每步实现的谓词为策略的规划方法,每个规划步是一个需实现的确定性谓词,该规划步称为谓词规划步.

显然,现在所公布的规划器[1,2]都是采用动作规划方法.在规划过程中,以确定每步的规划动作为研究目标,每个规划步是一个规划动作,由规划动作来实现相应的目标.在确定具体规划动作时,会采用各种评估策略(或启发函数)来评价所有“可选”动作的优劣,从而做出最优的动作选择.

规划器 StepByStep 采用谓词规划方法.在规划过程中,以确定每步所需实现的谓词为其研究目标.对当前规划步的谓词,我们确定实现它的可选规划动作,但无须确定用哪个具体的规划动作来实现,保留规划动作序列的多样性.所以,StepByStep 的规划过程是依次列出实现谓词的过程,而不是依次列出执行动作的过程.

本文在不引起混淆的情况下,将谓词规划步简称为规划步.

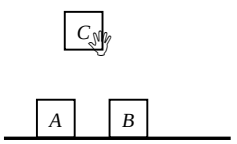

Fig.17 A mid-state for one planning problem

图 17 一个规划问题的中间规划状态

例如,BlocksWorld 领域中 Sussman 奇异问题的一个中间规划状态如图 17 所示.在该状态下,假设当前的规划步为(arm-empty),实现该规划步的可选动作有 3 个: (Stack *C A*),(Stack *C B*)和(Putdown *C*).

这时,动作规划方法可通过启发函数来评估这 3 个可选动作,当然可能会确定动作(Putdown *C*)为“最优”而被选择,也可能会选择其他动作而带来多余的规划动作.利用启发函数

很难做到:在所有情况下都能确定最佳的规划动作,这就如同我们不可能凭经验来得到所有问题的最佳解一样.

在规划器 StepByStep 中,只需确定当前规划步是实现谓词(arm-empty),则这 3 个动作都可选,在当前规划步无需进一步确定用哪个具体动作来实现.

在随后的规划过程中,需要实现谓词(hodling *B*),且存在下面绝对阻碍规则:

$$(\text{on } y\ x)_{\{(\text{not (on } y\ x)),(\text{clear } x),(\text{not (holding } x))\}} =\backslash\backslash=> (\text{holding } x)$$

这时,可确定前面规划步(arm-empty)不能采用动作(Stack *C B*),但其他两个动作仍然可选.

同样,当要实现谓词(hodling *A*)时,也可确定不能采用动作(Stack *C A*)来实现谓词(arm-empty).

至此才能最终确定只可用动作(Putdown *C*)来实现谓词(arm-empty).这种延迟确定动作的方法无须改变前面的任何规划步,因为前面的规划步是在谓词(arm-empty)为真的前提下进行的,而不会涉及其他规划步.

下面用 BlocksWorld 领域的一个简单规划问题来说明:延迟确定规划动作方法的主要思想.

在如 18 所示的规划问题中,T_0={(clear *A*),(ontable *A*),(clear *B*) (ontable *B*) (holding *C*)},T_G={(on *B C*)}.

在初始状态 T_0 下,由目标(on *B C*)的规划树得到可实现的规划步(arm-empty).该规划步有 3 个可选动作来实现,如图 19(1)所示.该规划步之后的规划状态为 T_1.

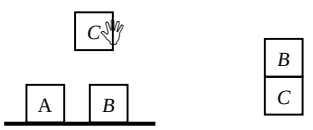

Fig.18 An example of BlocksWorld

图 18 一个简单的 BlocksWorld 规划问题

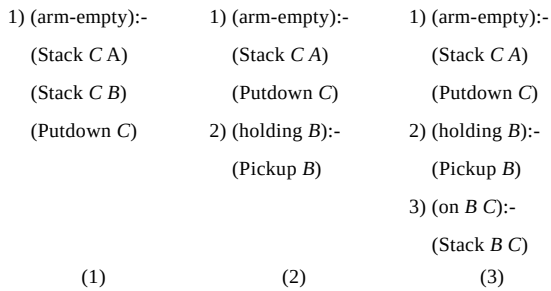

Fig.19 All steps of Fig.18 by StepByStep

图 19 StepByStep 求解图 18 的规划求解过程示意图

在规划状态 T_1 下,谓词(clear *C*)和(arm-empty)为真,有些谓词的真假值将变为不确定,如:(clear *A*),若使用动作(Stack *C A*),则谓词(clear *A*)在规划状态 T_1 为假,否则,该谓词仍然为真.

在规划状态 T_1 下,由(on *B C*)的规划树得到可实现规划步(holding *B*),该规划步之后的规划状态为 T_2.

由于存在直接阻碍关系:(not (clear *x*)) –\–> (holding *x*),即:没有(clear *x*)就直接阻碍谓词(holding *x*)的实现.也就是说,要实现规划步(holding *B*)就必须要先有谓词(clear *B*)为真,所以,必须删除规划步(arm-empty)中的可选动作(Stack *C B*).如图 19(2)所示.

在规划状态 T_2 下,由(on *B C*)的规划树得到规划步(on *B C*),该规划步之后的规划状态为 T_3.如图 19(3)所示.

至此,在规划状态 T_3 下,(on *B C*)的规划树仅含根结点,所以,StepByStep 的求解过程结束.

这时,在规划求解过程中得到 3 个谓词规划步:(arm-empty),(holding *B*)和(on *B C*).这 3 个谓词规划步可由两组动作序列来实现,它们所实现的目标状态如图 20(1)和图 20(2)所示.

(1) (Stack *C A*),(Pickup *B*),(Stack *B C*),

(2) (Putdown *C*),(Pickup *B*),(Stack *B C*).

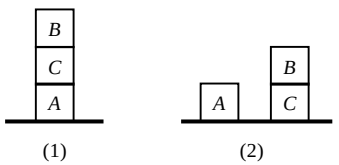

Fig.20 Two valid goal states for Fig.19

图 20 图 19 的两个合理的目标状态示意图

由此可知,一个谓词规划步序列可有多组动作序列来实现,且动作序列中的动作个数与规划步的步数是相等的,因为每个规划步只需选择一个规划动作来实现.

有关延迟确定规划动作的形式化描述及其相关证明将另文讨论,本文只给出其主要思想和求解步骤.

4.4 规划动作序列的多样性

由上节的叙述可知:一个规划步序列可对应多个步数相同的动作序列.这正反映了实现目标状态的动作序列的多样性.在确定规划步的可选动作集时,目前我们保留了一个规划步的所有可选动作.这样做的好处是可能得到最优解,但保留这些可能性会大量增加规划状态中的不确定性谓词,降低目标规划树的生成效率,也会增加规划步的处理时间.对一些大规模的规划问题,我们可能需要保留一个适当的可选动作数目,而不是保留所有的可选动作数目.比如,对每个规划步保留 5~10 个可选动作就可能保证会得到一个较佳的动作序列,而无须追求所谓最佳的动作序列.

对一个规划步需保留多少种可选动作比较合适的问题是一个看似简单,但实际上是一个比较复杂的问题.它就像在人工智能中对一个问题需要“向前思考多少步”的问题一样,向前思考的步数越多,做出的决策就越可靠,所花的时间也就越多.同样,一个规划步保留的可选动作个数越多,实现规划步的可能性就越大,得到最优动作序列的可能性也就越大.

下面用如图 21 所示的例子来说明我们的思考.

$$T_0=\{(\text{holding } A),(\text{clear } B_1),(\text{ontable } B_1),(\text{clear } B_2),(\text{ontable } B_2),\ldots,(\text{clear } B_{100}),(\text{ontable } B_{100})\},$$

$$T_G=\{(\text{arm-empty})\}.$$

目前,我们可给出 101 种实现目标状态的规划动作:(Putdown A)或(Stack A B_i),i=1..100.

如果 T_G={(on B_1 B_2)},那么,我们会得到下面 3 个规划步,共有 99 种规划动作序列.

(1) (arm-empty):- (Putdown A),(Stack A B_i),i=3..100 // 共有 99 个可选动作

(2) (holding B_1):- (Pickup B_1)

(3) (on B_1 B_2):- (Stack B_1 B_2)

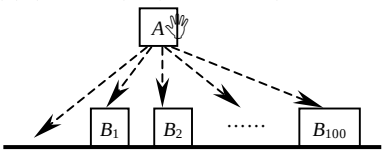

Fig.21 An planning problem of BlocksWorld

图 21 BlocksWorld 领域的一个规划问题

对一个规划问题,我们可能没有必要知道所有可能的解决方案,但若一个规划步保留的可选动作个数偏少,则会因前面的错误动作而导致后面需增加一些额外的规划步.比如:若规划步(arm-empty)的可选动作保留偏少,在只剩最后一个动作时,就必须执行该动作.当然,这时就可能使某积木块放错了地方.于是,在后面的规划过程中,还需要添加新的规划步来把它放到合适的位置.

如何在求解效率和规划质量之间取一个较好的平衡,是一个值得深入研究的问题.但从直观上来看,该问题更偏重于经验的积累和总结,所以,我们在以后的具体实验过程中将进行相关参数的设置和分析.

4.5 规划策略的算法描述

规划器 StepByStep 规划策略是在领域知识指导下的规划求解.在规划过程中以谓词规划步为规划主体,每个规划步可有多个可选动作,但在规划过程中不确定规划步所采用的具体动作,除非规划步只有一个可选动作.

规划器 StepByStep 规划策略的主要思想描述如下:

(1) 利用自动提取策略提取出基本的领域知识,利用动作之间相互作用获得复合的领域知识[19,20];

(2) 假设当前规划问题有 m 个子目标:$Pd_1,Pd_2,\ldots,Pd_m$;

(3) 利用排斥性阻碍规则对目标状态中的所有子目标进行排序[20];

(4) 若子目标序列中存在循环,则输出目标状态不可实现,并结束规划求解;

(5) 在初始状态下构造子目标 Pd_i 的规划树 PT_i(i=1..m),并获得其可实现的候选谓词集;

(6) 若存在一个 Pd_i 为假且相应的规划树 PT_i 仅含根结点($i\in\{1,2,...,m\}$),则输出目标状态不可实现,并结束规划操作;

(7) 置所有规划子目标为就绪状态;

(8) while (存在未完成的规划子目标) {

(8.1) 收集所有子目标规划树中可实现的候选谓词集;

(8.2) 利用领域知识从候选谓词集删除一些候选谓词;

(8.3) 对候选谓词集中剩下的每个谓词 *Pd*,做:若规划步 *Pd* 的所有前提条件都满足,则添加规划步 *Pd*,并根据需要来修改先前规划步中的可选动作,用所有可选动作的公共效果来改变当前规划状态,否则置该规划步所在的子目标为等待状态;

(8.4) 在新的规划状态下,若处于等待状态的子目标所需的资源已存在,则置其为就绪状态;

(8.5) 在新的规划状态下,更新所有就绪状态的规划树,并重新获得其可实现的候选谓词集;

}

(9) 目标状态中的所有子目标都已实现,依次输出每个规划步,选择其一个可选规划动作作为其实现的动作,并对后续的规划步作相应的修改.

步骤 8 是 StepByStep 规划策略的主要部分,也是规划策略中最具特色之处.具体表现在以下 3 个方面:

(1) 规划求解过程中,用所有规划树来确定待实现的谓词,不确定实现谓词的具体动作,保留所有可选动作.

(2) 规划求解过程中,在所有规划树的条件下,求出一个规划步或多个“并行”规划步,并改变当前规划状态.此后,在新的规划状态下更新所有“就绪”状态的规划树,在所有新规划树的条件下,又只确定一个规划步,如此反复,直至所有子目标都实现为止.显然,规划器在求解过程中是一个规划步、一个规划步地来进行求解的,这就是规划器命名为 StepByStep 的原因,其含义就是 Step-By-Step.

(3) 每个规划步都是在整个规划问题的全局环境下确定的.当前规划步取自一个规划树,下一个规划步可取自另一个规划树,规划步可以在不同的规划树之间跳跃.它不是采用确定一个子目标就一定实现它的策略,而是采用确定一个子目标只做一个规划步的策略.前者在实现选定子目标的过程中无法考虑其他子目标的实现,而后者在完成一个规划步后,将在新的状态下选择下一个规划步,这样会更有利于整个目标状态的实现.

4.6 规划器StepByStep的求解效率分析

我们在 Linux 环境下用 C 语言实现了规划器 StepByStep,该规划器的结构如图 2 所示.为了比较规划器的规划效率,我们在有关智能规划的研究网站下载了若干个规划器,如:AltAlt,Blackbox,IPP,HSP,MIPS,LPG 和 LPG-td[23−28],其中有些规划器在以往的规划大赛上取得了优异的成绩.

(一) 实验环境和测试的规划领域

硬件环境: Dell Latitude D600,Pentium M 1.8G,内存 512M,Windows XP Professional.

模拟环境: VMware Workstation 5.5,Red Hat Linux 7.32,KDevelop(开发环境).

规划领域: 从规划研究的网站下载的基准规划领域及其相应的规划问题.

(二) 规划领域的领域知识

规划器 StepByStep 在求解过程中先提取规划领域的基本知识,然后根据规划动作之间的相互作用进行逻辑,获得规划领域的复合知识.所选规划领域的领域知识个数见表 1,领域知识的具体形式从略.

Table 1 The number of domain rules extracting from planning domains

表 1 从规划领域中获取的领域知识个数

Planning domain	Basic laws of planning domain			Compound laws of planning domain	
	Direct concomitant rule	Conditional concomitant rule	Direct obstructive rule	Indirect obstructive rule	Absolute obstructive rule
BlacksWorld	22	8	11	9	33
Gripper	14	0	13	16	32
Logistics	2	1	3	0	3

(三) 规划器的规划效率分析

本文的测试以比较规划器种类为主要考量,所选的 3 个规划领域是具有一定代表性的经典规划领域.

(1) BlocksWorld 领域

该领域是规划研究常用的一个经典规划领域,它描述了一个机械臂堆放积木的 4 种操作.由于该领域只有

一个机械臂,任何操作都必须由该机械臂来完成,所以,动作之间具有很强的关联性,动作定义中的共性部分也非常多.因此,利用领域知识的自动提取策略能够获取较多的基本知识,利用动作之间的相互作用也能得到较多的复合知识,见表 1.

在表 1 中,Time 表示规划器的求解时间,时间单位为秒(s),Step 表示动作序列的步数,表格中的"--"表示某规划器在 30 分钟内未获得相应规划问题的解.

注:Time 表示规划器的求解时间,时间单位为秒(s),Step 表示动作序列的步数,表格中的"--"表示某规划器在 30 分钟内未获得相应规划问题的解.

对该领域的规划问题,我们知道,其规划状态空间将随其积木块个数的增加而呈指数增长.表 2 和表 3 所示的实验数据也反映了这种趋势.对一些小规模的规划问题,其他规划器还都能求出规划,但随着积木块个数的增加,这些规划器的求解时间和规划步数也都快速地增长,有的规划器甚至在 30 分钟内都无法完成求解过程.而规划器 StepByStep 对表 2 和表 3 中的所有规划问题都有很好的表现,不仅能在 1s 内完成表 3 中最大规模的规划问题,而且所得的规划步也非常少.

表 2 The planning performances for small-scale BlacksWorld problems

表 2 小规模 BlacksWorld 规划问题的求解效果列表

Problem	StepByStep		AltAlt		Blackbox		IPP	
	Time	Step	Time	Step	Time	Step	Time	Step
blocks-5	0.003	12	0.022	12	0.294	12	0.010	12
blocks-6	0.005	12	0.027	12	0.200	12	0.010	12
blocks-7	0.011	20	0.040	24	0.582	20	0.010	20
blocks-8	0.033	18	0.049	20	0.933	18	0.030	18
blocks-9	0.073	30	0.093	36	2.219	30	0.130	30
blocks-10	0.037	34	0.111	42	4.956	34	0.270	34
blocks-11	0.017	32	0.121	34	--	--	8.690	32
blocks-12	0.066	34	0.144	34	--	--	1.790	34
blocks-13	0.058	42	0.323	58	--	--	--	--
blocks-14	0.093	38	0.285	38	--	--	--	--
blocks-15	0.167	40	0.969	50	--	--	--	--
blocks-16	0.107	54	1.705	84	--	--	--	--
Problem	HSP		MIPS		LPG		LPG-td	
	Time	Step	Time	Step	Time	Step	Time	Step
blocks-5	0.001	18	0.050	16	0.060	22	0.020	22
blocks-6	0.010	24	0.100	12	0.040	12	0.010	22
blocks-7	0.001	22	0.080	24	0.100	42	0.040	54
blocks-8	0.010	26	0.130	22	0.050	38	0.050	40
blocks-9	0.050	48	12.600	38	0.280	96	0.070	70
blocks-10	0.050	46	1.040	40	0.840	138	0.080	56
blocks-11	0.090	54	--	--	0.290	98	0.100	60
blocks-12	0.650	58	21.660	44	0.760	198	0.170	88
blocks-13	0.580	64	--	--	1.150	184	0.270	90
blocks-14	3.420	68	--	--	2.590	108	0.200	84
blocks-15	1.390	70	--	--	8.990	180	0.330	110
blocks-16	0.900	86	--	--	10.890	246	0.250	94

规划器 StepByStep 对 BlocksWorld 领域的规划问题有如此优异表现的主要原因包括:

- 领域定义中隐含了足够的领域知识,我们通过领域知识提取策略把它们提取出来,并在求解过程中加以应用,如:子目标的排序和候选谓词规划的选择等;
- 该领域有唯一的独占性资源——机械臂,所有动作都必须由该机械臂来完成.机械臂的状态在整个规划中都是确定的,它只在"手空"和"手不空"(抓住某积木块)之间进行切换;
- 规划过程中,虽然对每个规划步都保留了所有的可选规划动作,增加了谓词状态的不确定性,但这些不确定性谓词对求解效率没有太大的影响,因为机械臂的状态始终是确定的.

(2) Gripper 领域

该领域是机器人在不同房间运送气球的规划领域,它定义了机器人的各种动作.由于动作只涉及机器人,所以,动作中的共性部分也比较多,我们可从动作定义中提取出较多的领域知识,见表 1.

该规划领域含有重复资源——机器人的左右手(或夹子),且左右手具有等价的功能.在需夹气球移动时,规

划器 StepByStep 不确定是用左夹子,还是用右夹子.这样,每个夹子都可能夹了气球,也都可能没夹.当需再夹一个气球时,还是可用左、右夹子来夹.处理这种谓词状态的“不确定性”将消耗更多的求解时间.

Table 3 The planning performances for large-scale BlacksWorld problems

表 3 较大规模 BlacksWorld 规划问题的求解性能对比

Problem	StepByStep		AltAlt		Blackbox		IPP	
	Time	Step	Time	Step	Time	Step	Time	Step
probblocks-17	0.190	58	6.870	82	--	--	--	--
probblocks-18	0.205	64	--	--	--	--	--	--
probblocks-19	0.347	62	4.038	78	--	--	--	--
probblocks-20	0.254	60	3.113	82	--	--	--	--
probblocks-21	0.371	78	--	--	--	--	--	--
probblocks-22	0.318	66	--	--	--	--	--	--
probblocks-23	0.388	74	6.056	106	--	--	--	--
probblocks-24	0.291	78	--	--	--	--	--	--
probblocks-25	0.231	82	--	--	--	--	--	--
probblocks-26	0.403	84	--	--	--	--	--	--
probblocks-27	0.831	84	--	--	--	--	--	--
probblocks-28	0.937	92	--	--	--	--	--	--
Problem	HSP		MIPS		LPG		LPG-td	
	Time	Step	Time	Step	Time	Step	Time	Step
probblocks-17	3.560	84	--	--	20.800	234	0.730	124
probblocks-18	3.200	94	--	--	35.190	400	0.800	124
probblocks-19	92.080	98	--	--	37.920	402	1.290	114
probblocks-20	107.720	112	--	--	50.110	382	0.790	160
probblocks-21	--	--	--	--	77.740	424	1.370	160
probblocks-22	--	--	--	--	171.020	382	8.700	202
probblocks-23	--	--	--	--	145.580	574	4.690	156
probblocks-24	--	--	--	--	324.910	816	2.740	148
probblocks-25	--	--	--	--	330.120	448	12.540	176
probblocks-26	--	--	--	--	419.800	476	18.360	224
probblocks-27	--	--	--	--	500.900	886	7.750	196
probblocks-28	--	--	--	--	1375.720	1596	11.290	212

由此可见,我们在求解过程中增加了谓词的不确定性,保留了规划动作的多样性.这种求解策略增加了规划过程的复杂度,降低了求解效率,但它也会减少所得的规划步数,表 4 所列的实验数据验证了该分析结论.StepByStep 规划器的求解时间要略多于其他规划器,但所求的规划步是最少的,甚至可计算出它们都是最优的.

Table 4 The planning performances for Gripper problems

表 4 Gripper 规划问题的求解性能对比

Problem	StepByStep		AltAlt		Blackbox		IPP	
	Time	Step	Time	Step	Time	Step	Time	Step
gripper02	0.003	6	0.050	7	0.010	6	0.010	6
gripper04	0.003	9	0.003	9	0.030	9	0.001	9
gripper06	0.008	15	0.010	15	0.650	15	0.040	15
gripper08	0.026	23	0.034	23	--	--	0.790	23
gripper15	0.116	45	0.298	45	--	--	--	--
gripper20	0.274	59	1.387	59	--	--	--	--
gripper25	0.486	69	0.818	69	--	--	--	--
gripper30	2.128	79	1.121	79	--	--	--	--
Problem	HSP		MIPS		LPG		LPG-td	
	Time	Step	Time	Step	Time	Step	Time	Step
gripper02	0.010	6	0.050	6	0.001	6	0.010	6
gripper04	0.001	9	0.040	11	0.010	9	0.010	11
gripper06	0.001	17	0.040	19	0.010	15	0.020	19
gripper08	0.001	29	0.040	31	0.010	23	0.030	31
gripper15	0.030	57	0.060	59	0.010	47	0.020	57
gripper20	0.100	77	0.070	79	0.020	67	0.040	77
gripper25	0.150	87	0.100	89	0.040	75	0.040	100
gripper30	0.360	97	0.210	99	0.050	88	0.090	124

(3) Logistics 领域

该领域是一个关于物流运输的规划领域,它规定:卡车只能在其所处的城市内移动,不能在城市之间移动;飞机可在有机场的城市之间移动,包裹(package)可由卡车或飞机来移动.在卡车和飞机这两类运输工具的动作

定义中很少有公共谓词,所以,从动作定义中只能提取出很少的领域知识,见表 1.

该规划领域含有较多的静态谓词,如:(package ?*obj*),(truck ?*truck*),(airplane ?*airplane*),(airport ?*airport*),(location ?*loc*),(in-city ?*obj* ?*city*)和(city ?*city*)等,仅有两个动态谓词:(at ?*obj* ?*loc*)和(in ?*obj*1 ?*obj*2),前者表示?*obj*(包裹,卡车或飞机)在位置?*loc* 处,后者表示包裹?*obj*1 在运输工具?*obj*2(卡车或飞机)之中.

由于没有足够的领域知识来指导,规划器 StepByStep 的求解时间比其他规划器要多,但在所求的规划步数方面有很好的表现.仅对两个规划问题 probLOGISTICS-11 和 probLOGISTICS-12,StepByStep 所得的规划步数要比 MIPS 多 1 步.除此之外,StepByStep 的规划质量都比其他规划器要好,见表 5.

Table 5 The planning performances for logistics problems

表 5 Logistics 规划问题的求解性能对比

Problem	StepByStep		AltAlt		Blackbox		IPP	
	Time	Step	Time	Step	Time	Step	Time	Step
probLOGISTICS-4	0.003	6	0.024	24	--	--	--	--
probLOGISTICS-5	0.006	17	0.025	19	--	--	--	--
probLOGISTICS-6	0.001	14	0.025	16	--	--	--	--
probLOGISTICS-7	0.017	46	0.062	54	--	--	--	--
probLOGISTICS-8	0.099	34	0.078	39	--	--	--	--
probLOGISTICS-9	0.122	44	0.052	50	--	--	--	--
probLOGISTICS-10	0.344	51	0.102	57	--	--	--	--
probLOGISTICS-11	0.412	58	0.106	63	--	--	--	--
probLOGISTICS-12	0.556	81	0.153	95	--	--	--	--
probLOGISTICS-13	7.754	84	0.276	93	--	--	--	--
probLOGISTICS-14	5.890	69	0.208	74	--	--	--	--
probLOGISTICS-15	8.282	72	0.280	89	--	--	--	--
Problem	HSP		MIPS		LPG		LPG-td	
	Time	Step	Time	Step	Time	Step	Time	Step
probLOGISTICS-4	0.001	26	0.080	17	0.010	21	0.060	25
probLOGISTICS-5	0.030	21	0.040	22	0.030	21	0.030	22
probLOGISTICS-6	0.010	16	0.040	14	0.010	15	0.010	17
probLOGISTICS-7	0.090	56	0.320	61	0.030	46	0.020	58
probLOGISTICS-8	0.080	40	0.570	39	0.030	38	0.020	42
probLOGISTICS-9	0.080	51	0.060	44	0.030	48	0.020	48
probLOGISTICS-10	0.200	59	0.140	53	0.210	49	0.080	54
probLOGISTICS-11	0.330	69	0.150	57	0.310	72	0.040	61
probLOGISTICS-12	0.300	98	0.220	80	0.130	81	0.050	89
probLOGISTICS-13	0.910	102	2.000	94	0.080	101	0.100	106
probLOGISTICS-14	0.810	82	0.550	72	0.040	75	0.080	84
probLOGISTICS-15	0.950	98	1.470	84	0.030	83	0.060	95

综合 8 个规划器对所选 3 个规划领域进行规划求解的实际表现,我们不难看出:规划器 StepByStep 在规划效率和规划质量上都有很好的表现,尤其是在规划质量上.除个别规划问题比 MIPS 多 1 步外,对其他规划问题,StepByStep 所得到的规划步数都是最少的.在求解效率方面,由于 StepByStep 目前采取保留所有可选规划动作的策略,所以,在求解过程中需消耗更多的时间.正如第 4.4 节所述,如果适当地减少可选动作,降低动作序列的多样性等,那么 StepByStep 所求的规划步数可能会增加,但规划效率也一定会有所提高.在规划效率和规划质量之间寻找一个平衡的折衷点,我们还需做进一步的分析研究.

5 结 论

本文简单介绍了规划器的一般结构和 StepByStep 规划器的内部结构,从宏观上看,两者具有相同的外观结构;但从求解策略上来看,规划器 StepByStep 有其独特之处,它利用从领域定义中提取出来的领域来指导规划求解,使规划策略具有更可靠的领域基础.

文中对 StepByStep 规划器内部所采用的方法给予了较详细的描述,用谓词知识树来提取领域知识,并进行领域知识的逻辑推导.在谓词知识树的基础上定义谓词规划树,并利用各种策略来提高规划树的生成效率.然后利用谓词规划树来设计 StepByStep 规划器的规划策略,最后用 8 个规划器对 3 个具有代表性的基准规划领域及其规划问题进行实际的求解实验,分析 StepByStep 规划器在规划步数和规划动作序列的多样性等方面的优势,

也分析了由此带来求解效率上的损失.实验数据表明,规划器 StepByStep 的规划策略对 3 个不同的规划领域都具有很好的指导作用,验证了领域知识在规划求解过程中的实际价值.

References:

[1] 2007. http://www.aiai.ed.ac.uk/links/planning.html

[2] 2007. http://www.csc.ncsu.edu/faculty/stamant/planning-resources.html

[3] International Planning Competition. 2007. http://zeus.ing.unibs.it/ipc-5/

[4] Jonsson A, Morris P, Muscettola N, Rajan K, Smith B. Planning in interplanetary space: Theory and practice. In: Proc.of the 4th AIPS-00. AAAI Press, 2000. 177−186.

[5] Green C. Application of theorem proving to problem solving. In: Proc. of the 1st Int'l Joint Conf. on AI. 1969. 219−239.

[6] Richard Fikes, Nils J. Nilsson: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 1971,2(3/4):189−208.

[7] Fox M, Long D. The automatic inference of state invariants in TIM. Journal of AI Research, 1998,9:367−421.

[8] Blum A, Furst M. Fast planning through planning graph analysis. Artificial Intelligence, 1997,90(1)2:281−300.

[9] Thiébaux S, Hoffmann J, Nebel B. In defense of PDDL axioms. Artificial Intelligence, 2005,168:38−69.

[10] Kvarnström J, Doherty P. TALplanner: A temporal logic based forward chaining planner. Annals of Mathematics and Artificial Intelligence (AMAI), 2001,30:119−169.

[11] Doherty P, Kvarnström J. TALplanner: A temporal logic based planner. AI Magazine, 2001.

[12] Kautz H, Selman B. BLACKBOX: A new approach to the application of theorem proving to problem solving. In: Workshop Planning as Combinatorial Search, AIPS-98, Pittsburgh, 1998. 58−60. http://citeseer.ist.psu.edu/kautz98blackbox.html

[13] Kautz H, Selman B. The role of domain-specific knowledge in the planning as satisfiability framework. In: Proc. of the 4th IJCAI Calif. Menlo Park. AAAI Press, 1998. 181−189. http://citeseer.ist.psu.edu/kautz98role.html

[14] Nguyen X, Kambhampati S, Nigenda RS. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. Artificial Intelligence Journal, 2002,135(1−2):73−123.

[15] Nigenda RS, Nguyen X, Kambhampati S. AltAlt: Combining the advantages of graphplan and heuristic state search. ASU Technical Report, 2000.

[16] Andersen HR. An introduction to binary decision diagrams. Technical Report, 1997. http://www.itu.dk/people/hra/bdd97-abstract.html

[17] Edelkamp S, Helmert M. On the implementation of MIPS. In: Proc. of the Artificial Intelligence Planning and Scheduling (AIPS), Workshop on Decision-Theoretic Planning. Breckenridge, 2000. 18−25. http://citeseer.ist.psu.edu/edelkamp00 implementation.html

[18] Edelkamp S, Helmert M. The model checking integrated planning system. AI-Magazine (AIMAG), 2001,67−71.

[19] Wu XJ, Jiang YF, Ling YB. Strategy of extracting domain knowledge for STRIPS world. Journal of Software, 2007,18(3):490−504. (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/490.htm

[20] Wu XJ, Jiang YF, Ling YB. Researches on relations of effect of action for STRIPS domain, Accepted by Journal of Software, 2007,18(6):1328−1349 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1328.htm

[21] Drew McDermott, The 1998 AI planning systems competition. AI Magazine, 2000,21(2):35−55.

[22] Alfonso Gerevini, Ivan Serina, LPG: A Planner based on local search for planning graphs for action costs. In: Proc. of the 6th Int'l Conf. on Artificial Intelligence Planning and Scheduling (AIPS 2002). AAAI Press, 2002.

[23] AltAlt. 2007. http://rakaposhi.eas.asu.edu/altweb/altalt.html

[24] BlockBox. 2007. http://www.cs.rochester.edu/u/kautz/satplan/blackbox/index.html

[25] IPP. 2007. http://www.informatik.uni-freiburg.de/~koehler/ipp.html

[26] HSP. 2007. http://scom.hud.ac.uk/planet/repository/heuristic.html

[27] MIPS. 2007. http://ls5-web.cs.uni-dortmund.de/~edelkamp/mips/

[28] LPG/LPG-td. 2007. http://zeus.ing.unibs.it/lpg/

附中文参考文献:

[19] 吴向军,姜云飞,凌应标.基于 STRIPS 的领域知识提取策略.软件学报,2007,18(3):490−504. http://www.jos.org.cn/1000-9825/18/490.htm

[20] 吴向军,姜云飞,凌应标.STRIPS 规划领域中动作效果关系的研究.软件学报,2007,18(6):1328−1349. http://www.jos.org.cn/1000-9825/18/1328.htm.

附录 A. Sussman 奇异问题的描述

(define (problem Sussman-Anomaly)
(:domain BlocksWorld)
(:objects *A B C*)
(:init (clear *C*) (on *C A*) (ontable *A*) (clear *B*) (ontable *B*) (arm-empty))
(:goal (and (on *A B*) (on *B C*)))
)

附录 B. Sussman 奇异问题的规划树

在 Sussman 奇异问题中,子目标(on *A B*)和(on *B C*)在初始状态下的规划树如图 22 和图 23 所示.

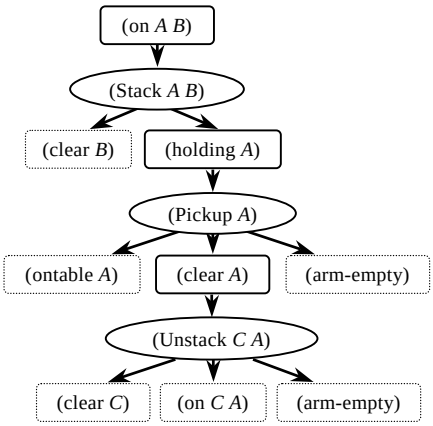

Fig.22 Planning tree of sub-goal (on *A B*)

图 22 子目标(on *A B*)的规划树

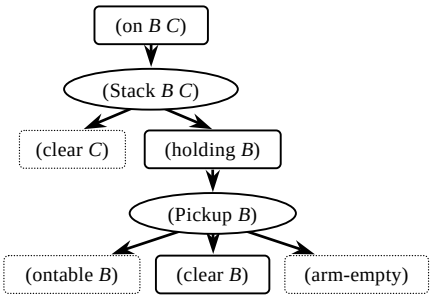

Fig.23 Planning tree of sub-goal (on *B C*)

图 23 子目标(on *B C*)的规划树

在子目标(on *A B*)和(on *B C*)的规划树中,当前可实现的规划步分别为(clear *A*)和(holding *B*),所以,在初始状态下,可实现的候选谓词集为{(clear *A*),(holding *B*)}.

吴向军(1965－),男,广东广州人,副教授,主要研究领域为人工智能,算法设计,网络应用.

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,基于模型诊断.

凌应标(1965－),男,博士,副教授,主要研究领域为智能规划,演化计算,智能优化.