软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(8):1743−1760 [doi: 10.13328/j.cnki.jos.004513]

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

面向结构的基于学习的规划方法*

陈蔼祥[1], 姜云飞[3], 柴啸龙[1], 边 芮[2], 陈清亮[4]

[1](广东财经大学 数学与统计学院,广东 广州 510320)

[2](广东财经大学 公共管理学院,广东 广州 510320)

[3](中山大学 软件研究所,广东 广州 510275)

[4](暨南大学 计算机系,广东 广州 510632)

通讯作者: 柴啸龙, E-mail: chaixiaolongok@163.com

摘 要: 近年来,规划中的学习问题重新受到了关注.如何通过学习机制改善现有规划器,使其能够可靠而令人信服地超越现有非学习的规划器的能力,仍然是一个尚未解决的难题.提出了面向规划问题和解的结构的基于学习的规划技术.该方法将先验知识表示成"子问题-规划片段"的形式.每次规划器成功找到解以后,根据问题的初始状态和目标状态,构造规划对象的初始子状态和目标子状态,构成子问题,并从规划解中抽取该子问题对应的规划片段.这些先验知识将被唯一记录并保存成先验知识库.新问题的求解首先从先验知识库中检索与当前求解问题相关的先验知识;然后,将这些先验知识经过例化、合并步骤后编码成句子;最后,将这些句子连同问题编码得到的句子作为SAT求解器的输入,实现最终解的确定.实验使用了IPC中的基准测试例子进行测试.实验结果表明,SOLP算法求解速度与传统非学习的规划器相比具有明显优势,最佳情况下可达约80%的效率提升.

关键词: 问题结构;解结构;规划片段;结构知识学习

中图法分类号: TP181

中文引用格式: 陈蔼祥,姜云飞,柴啸龙,边芮,陈清亮.面向结构的基于学习的规划方法.软件学报,2014,25(8):1743−1760. http://www.jos.org.cn/1000-9825/4513.htm

英文引用格式: Chen AX, Jiang YF, Chai XL, Bian R, Chen QL. Structure-Oriented learning-based planning method. Ruan Jian Xue Bao/Journal of Software, 2014,25(8):1743−1760 (in Chinese). http://www.jos.org.cn/1000-9825/4513.htm

Structure-Oriented Learning-Based Planning Method

CHEN Ai-Xiang[1], JIANG Yun-Fei[3], CHAI Xiao-Long[1], BIAN Rui[2], CHEN Qing-Liang[4]

[1](School of Mathematics and Statistics, Guangdong University of Finance and Economics, Guangzhou 510320, China)

[2](School of Public management, Guangdong University of Finance and Economics, Guangzhou 510320, China)

[3](Software Research Institute, Sun Yat-Sen University, Guangzhou 510275, China)

[4](Department of Computer, Ji'nan University, Guangzhou 510632, China)

Corresponding author: CHAI Xiao-Long, E-mail: chaixiaolongok@163.com

Abstract: The goal of reliably outperforming non-learning planners via learning is still to be achieved. A novel structure-oriented learning-based planning method (SOLP) is presented. SOLP anaylyses the structure knowledge, decomposes the planning problem into initial sub-state and goal sub-state, its solution into plan fragment, when planner finds out a solution successfully. The structure knowledge from previous experiment, or prior knowledge, will be saved in domain. When encountering new problem, SOLP firstly recalls

* 基金项目: 国家重点基础研究发展计划(973)(2005CB321902, 2010CB328103); 国家自然科学基金(60773201, 61003056);广东省自然科学基金(10451032001006140); 广州市科技和信息化局应用基础研究计划(2010Y1-C641); 广东省教育厅高校优秀青年创新人才培育项目(LYM10081, LYM_0065); 中央高校基本科研业务费专项资金(21612414); 广东省教育厅科技创新项目(2013kjcx0086); 广东财经大学自然科学研究项目(11BS52001)

收稿时间: 2013-05-27; 修改时间: 2013-07-30; 定稿时间: 2013-10-11

the prior problem structure equivalent or similar to the current problem and the corresponding plan fragment from the domain file, then instantiates the learned prior knowledge as ground knowledge, and finally, encodes the ground knowledge as a satisfiability clause. These clauses, together with the set of clauses from the problem, form the input of the algorithm. SOLP calls the SAT Solver to determine the final solution. An experiment is conducted to test the algorithm in several different domains from IPC to demonstrate the efficiency and effectiveness of the new approach. The results show that, the speed of SOLP has obvious advantage than that of non-learning planner, with up to 80% improvement in extreme case.

Key words: problem structure; solution structure; plan fragment; structure knowledge learning

自动规划系统要求在给定动作模型下能够自动构造动作序列,完成特定任务.现有的规划器一般不具备学习和记忆能力,每次问题求解,均是从零开始构造规划解.真正的智能系统应该具备从“经验”中学习知识并变“聪明”的能力.为了鼓励规划中的学习问题的研究,第 6 届国际智能规划大赛开辟了学习轨道的比赛,许多研究者提出了许多不同的知识表示形式,比如搜索策略[1−4]、宏动作序列[5−7]、子目标分解知识[8−10]、启发式函数[11]等.PbP[12]则是学习相应知识指导系统,以最有效地组合领域无关规划器进行求解的规划系统.

一般地,学习的知识只在相似的问题之间才有效.Nebel 等人[13]给出了命题匹配概念下的相似问题评价方法.在他们的命题匹配方法下,两问题是否相似等价于图论中的子图同构问题,而子图同构问题是一个众所周知的 NP-hard,从而得出了从理论角度看,学习无法提高规划效率这一反直觉的结论.目前在规划中的学习领域取得的结果似乎证实了 Nebel 等人的上述结论:与非学习的规划器相比,学习机制并不能提高系统的整体性能.

我们认为:以上这些方法均未能把握规划中的学习问题的本质,从而未能提出一个恰当的知识表示形式,导致知识难以被有效学习或学习得到的知识的使用代价过高.这导致学习机制的使用削弱了系统总体性能.

人脑学习能力的有效性主要在于能够记住某些关键知识,并能在求解新问题时有选择性地回忆相关有用知识,从而加速问题求解.

图 1 是涉及到不同积木块对象的两个不同规划问题.显然,对于第 1 个规划问题,我们容易找到其解为 $\langle pickup(A),stack(A,B),pickup(C),stack(C,A)\rangle$.对于第 2 个规划问题,容易发现,只要将第 1 个规划问题的解稍作调整,$A/a,B/b,C/c$,很容易就可以得到第 2 个规划问题的解:$pickup(a),stack(a,b),pickup(c),stack(c,a)$.这两个问题均要求将散落在桌面上的积木块按照特定的顺序要求堆成一堆,这使得他们的解具有相同的结构形式 *pickup*,*stack*,*pickup*,*stack*.

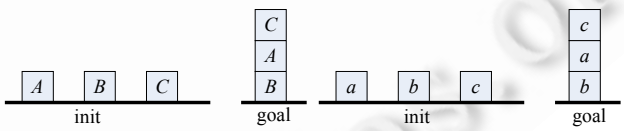


Fig.1 Two problems with the same structure

图 1 结构相同的两个不同问题

图 2 中的 3 个规划问题要求分别将 3 个积木块按指定的顺序堆成一堆.易见,这 3 个问题的解分别为

$\langle pickup(A),stack(A,B),pickup(C),stack(C,A)\rangle$,$\langle pickup(B),stack(B,A),pickup(C),stack(C,B)\rangle$,

$\langle pickup(B),stack(B,C),pickup(A),stack(A,B)\rangle$.

这些解的结构均是 *pickup*,*stack*,*pickup*,*stack* 的形式.

更进一步地,无论是图 1 还是图 2 中的问题,人脑都能很快地识别以下两类子问题:(1) 将摆在桌面上的积木块压在某些东西上面;(2) 摆放在桌面上的积木块被某些东西夹住.同样地,人脑很容易记住这两个子问题的解决办法:对于问题(1),要首先从桌面上拿起该积木($pickup_1$),然后将其堆放到合适的位置($stack_1$),即 $pickup_1$,$stack_1$;对于问题(2),首先要从桌面上拿起该积木($pickup_1$),然后把它堆放在某个地方($stack_1$),最后用其他积木将它压住($stack_2$),即$\langle pickup_1,stack_1,stack_2\rangle$.

传统的非学习规划器在求解上述类似的规划问题时,无法“记住”曾经完成过的类似任务,不能在不同的规划问题之间进行求解经验的共享/借鉴.传统规划器这种不带学习能力的工作方式极大地影响了规划器的求解

效率.

我们的目标主要是寻找新的学习方法,使得能在不同问题之间重用知识,并通过学习机制的使用切实提高规划系统的整体效率.我们的方法工作在两种状态下:学习状态和规划状态.算法的输入包括:(1) 对象集合;(2) 规划问题;(3) 规划解(该输入非空表明算法工作在学习状态,否则工作在规划状态);(4) 先验知识列表(每一先验知识由子问题-规划片段构成).我们的算法称为面向结构的基于学习的规划系统 SOLP(structure orienting learning-based planning system).若算法工作在学习状态,则其输出为学习到的先验知识列表;否则,算法处于规划状态,输出结果为规划解.

与以前的方法相比,本文工作的主要贡献在于:

1) 提出了一种新的先验知识表示形式;
2) 在我们的先验知识表示框架下,给出了一种新的能够在不同问题(这些问题结构甚至可以完全不同)之间共享/重用先验知识的方法;
3) 提出了一种能够根据学习到的先验知识重构/组合(部分)规划的方法.

图 3 是 SOLP 的总体结构.SOLP 大体分成 4 个步骤:

- 第 1 步(如果算法工作在学习状态),根据规划对象从规划问题中抽取该对象的子问题(图 3 中的“1?”),并从规划解中抽取规划片段(图 3 中“2?”),子问题和规划片段构成先验知识保存到领域描述文件(图 3 中的“4?”).
- 第 2 步,当求解新问题时,根据新问题对象构造子问题,并根据对象子问题从先验知识库中检索匹配的先验知识(图 3 中的“3?”).
- 第 3 步,检索到的先验知识将被例化后合并编码成可满足句子 C_μ.
- 第 4 步,这些代表先验知识的句子 C_μ连同问题编码得到的子句 C_{prob},作为 SAT Solver 的输入进行求解,并将最终结果转化成最终规划解.

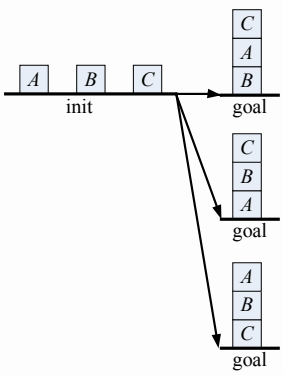


Fig.2 Three problems with similar structure

图 2 结构相似的 3 个不同问题

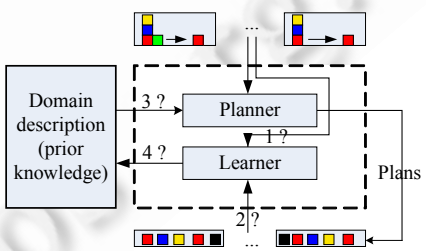


Fig.3 Architecture of SOLP

图 3 SOLP 体系结构

我们将对规划大赛中 STRIPS 格式的标准测试例子进行测试,以表明 SOLP 确实比传统的领域无关的非学习规划器具有更强的求解能力.目前,SOLP 尚不能处理诸如含资源约束、偏好约束、派生谓词、间隔保持约束等特性的问题.

本文第 1 节进行符号约定,并给出描述本文方法所需要的相关定义.第 2 节给出本文学习问题和带先验知识的规划问题的形式定义.第 3 节介绍本文的 SOLP 算法.该算法主要分成以下几个步骤:1) 先验知识的提取;2) 先验知识的检索、例化;3) 将先验知识合并、编码成可满足句子集;4) 带先验知识的基于 SAT 方法的规划求解.第 4 节从理论上对 SOLP 的相关性质进行分析.第 5 节给出我们的实验结果.最后总结并给出未来进一步的研究方向.

1 符号约定及相关定义

为了描述“子问题-规划片段”形式的先验知识,我们需要用到性质、动作片段等概念.最初提出性质这一概念的是 Fox 等人[14,15].性质ρ用某对象在该谓词中出现的位置作为该谓词的下标进行表示,n 元谓词共有 n 个性质,$\rho^1,\rho^2,\ldots,\rho^n$.性质集合用$\Omega$表示.$s$ 表示状态,o 表示规划对象,$ss=s_\rho^o$ 表示对象 o 在状态 s 中拥有的性质集合,称 ss 为对象 o 的子状态.ζ表示一个谓词,p 表示一个命题.ζ_i^o 表示谓词ζ的第 i 个参数为 o 的部分例化谓词.$ss=s_\zeta^o$ 表示被 o 部分例化的子状态.$ss_p=s_\rho^o$ 表示状态 s 下 o 的命题子状态.

类似于性质的定义,动作片段α由某对象在该动作模型中出现的位置作为该动作名称的下标表示,n 个参数的动作定义了 n 个动作片段$\alpha^1,\alpha^2,\ldots,\alpha^n$.有时为了突出规划对象,又用 $\alpha=\alpha_F^o$ 表示某实例动作 a 关于对象 o 的动作片段(α_F^o 表示对象 o 的 F 投影,这里,“F”代表“动作片段”的含义).动作片段集用Φ表示.m_i^o 表示领域 D 动作模型 m 的第 i 个参数被 o 例化后得到的部分例化动作模型.

Mali 研究了基于可满足框架下的规划合并和规划重用[16].类似地,为了能够将先验知识在可满足框架下进行合并和重用,我们需要用到以下符号:

P_i 表示规划步,a_j 表示基动作.U 表示领域中所有命题 u_j 的集合,$u_j(t)$表示 u_j 在时间步 t 成立.ϕ表示空动作.A 是包含空动作ϕ的基动作集.ϕ在时间步 j 成立,表示时间步 j 无其他非空动作.也就是说,空动作ϕ与其他非空动作互斥.(P_i=a_j)表示 $step\rightarrow action$ 映射,或称将 a_j 绑定到规划步 P_i.$a_i(j)$表示动作 a_i 在时间步 j 执行(a_i 的前件在时间步 j 成立,其效果在时间步(j+1)成立).$P_i\prec P_j$ 表示 P_i 先于 P_j.k_i 表示第 ith 条先验知识的动作片段数,m 表示需要合并的先验知识数.a_{ij} 表示来自 ith($i\in[1,m]$)先验知识 jth($i\in[1,k_i]$).K 表示 m 条先验知识需要合并的动作片段总数.

2 问题定义

经典规划问题表示成 $Prob=(\Sigma,I,G)$,其中,$\Sigma=(S,M,\delta)$表示规划领域,S 表示状态集合,M 表示动作模型集合,δ 表示确定型状态转移函数$\delta:S\times M\rightarrow S$,$I$ 表示初始状态,G 表示目标状态.动作模型定义为动作头部连同前提条件和效果,其中,动作头部由动作名称及零个或多个参数构成.规划(解)是能使初始状态 I 转换成目标状态 G 的动作序列$\langle a_0,a_1,\ldots,a_n\rangle$.

本文的学习问题可被描述为:给定规划问题 $Prob$,规划对象集 O,规划(解)Sol.作为学习的结果,SOLP(工作在学习状态)将输出先验知识列表ψ.

本文的带先验知识的规划问题可被描述为:给定规划问题 $Prob$,先验知识列表ψ.作为规划的结果,SOLP(工作在规划状态)输出最终解.

定义 1(状态 s 的结构). 任一合法状态 s,可表示成类似 $s=(o_1ss_1+o_2ss_2+\ldots+(o_i+o_j)ss_i+\ldots+o_nss_n)$的形式,其中,$o_1,o_2,\ldots,o_n$ 表示 s 中的对象;ss_i 表示对象 o_i 在状态 s 下所拥有的性质,即 $ss_i=s_\rho^{o_i}\in SS$;项$(o_i+o_j)ss_i$ 表示在状态 s 下,对象 o_i,o_j 拥有公共子状态 ss_i,称这些互异的子状态集$\{ss_1,\ldots,ss_i,\ldots,ss_n\}$构成了状态 s 的结构,记为Θ_s.

例如,图 2 中第一个问题的初试状态 I 可表示成$((A+B+C)\langle ontable_1,clear_1\rangle)$,因此,初试状态 I 的结构为$\Theta_I=\langle ontable_1,clear_1\rangle$.类似地,该问题的目标状态 G 可表示成$(C\langle on_1,clear_1\rangle+A\langle on_1,on_2\rangle+B\langle ontable_1,on_2\rangle)$,目标状态 G 的结构为$\Theta_G=\langle on_1,clear_1\rangle+\langle on_1,on_2\rangle+\langle ontable_1,on_2\rangle$.

定义 2(子问题(sub-problem)). 给定规划问题 $Prob$,类型为 t 的对象 o 的子问题 $sp(o)=\langle t,o,I_\rho,G_\rho\rangle$.

若不考虑具体对象,则忽略参数 o,而简记为 $sp=\langle t,I_\rho,G_\rho\rangle$.$sp_\rho=\langle t,I_\rho,G_\rho\rangle$,$sp_p=\langle t,I_p,G_p\rangle$分别表示谓词子问题和命题子问题.

例如,图 2 中第 1 个问题中关于积木块 A 的子问题 $sp=\langle block,A,\langle ontable_1,clear_1\rangle,\langle on_1,on_2\rangle\rangle$.

定义 3(子问题等价). 两个子问题 $sp_i=\langle t_i,o_i,I_\rho^{o_i},G_\rho^{o_i}\rangle,sp_j=\langle t_j,o_j,I_\rho^{o_j},G_\rho^{o_j}\rangle$ 相等当且仅当:

- $t_i=t_j$:o_i,o_j 是同一类型对象;
- $I_\rho^{o_i},I_\rho^{o_j}$:$o_i,o_j$ 在状态 I 中具有共同的性质;

- $G_{\rho}^{o_i}, G_{\rho}^{o_j}$:o_i,o_j 在状态 G 中具有共同的性质.

在子问题等价的基础上,我们可以刻画规划问题的相似程度.

定义 4(问题的相似度). 给定 $Prob$,该问题涉及到 $O=\{o_1,o_2,\ldots,o_n\}$这 n 个规划对象,另一规划问题 $Prob'$涉及到 $O'=\{o_1,o_2,\ldots,o_m\}$这 m 个规划对象.若两问题共有 k 个子问题相等,即, $sp(o_{i1})=sp(o'_{j1}),\ldots,sp(o_{ik})=sp(o'_{jk})$, 则两问题相似度定义为 $Sim(Prob,Prob')=\dfrac{k}{\max(m,n)}$.

定义 5(问题的结构). 任意规划问题 $Prob$ 总可以由 SP 中子问题表示成 $o_1sp_1+o_2sp_2+\ldots+(o_i+o_j)sp_i+\ldots+o_nsp_n$ 的形式,其中,sp_i 表示 SP 中子问题,项$(o_i+o_j)sp_i$ 表示对象 o_i,o_j 具有相同的子问题.这些互异的子问题集 $sp_1,sp_2,\ldots,sp_i,\ldots,sp_n$ 构成了 $Prob$ 的结构性知识,用符号$\Theta_{Prob}=\{sp_1,sp_2,\ldots,sp_i,\ldots,sp_n\}$表示.

例如,图 1 中第 1 个问题,积木块 A,B,C 对应的子问题分别为 $sp_A=\langle block,A,\langle ontable_1,clear_1\rangle,\langle on_1,on_2\rangle\rangle$,$sp_B=\langle block,B,\langle ontable_1,on_2\rangle,\langle on_1,clear_1\rangle\rangle$,$sp_C=\langle block,C,\langle on_1,clear_1\rangle,\langle on_1,on_2\rangle\rangle$,该问题的结构为 $\Theta_{Prob}=\{sp_A,sp_B,sp_C\}$. 显然,图 1 和图 2 中的 5 个问题是结构等价的问题,因为它们都有$\Theta_{Prob}=\{sp_A,sp_B,sp_C\}$这一共同的结构.

定义 6(规划片段(plan fragment)). 所谓规划片段,是二元组$\pi=\langle\Phi,R_\alpha\rangle$,其中,$\Phi=\{\alpha_1,\alpha_2,\ldots,\alpha_k\}$为动作片段集,$R_\alpha$为定义在$\Phi$上形如$\alpha_i\prec\alpha_j$($\alpha_i$ 先于α_j)的动作片段关系集.

定义 7(子问题解片段). 若规划片段π可将子问题 sp 中的初始状态转换为目标子状态,则称π为 sp 的规划片段解.

例如,图 1 中第 1 个问题的积木块 A 的子问题 $sp_A=\langle block,A,\langle ontable_1,clear_1\rangle,\langle on_1,on_2\rangle\rangle$对应的解片段为

$$pickup_1\prec stack_1\prec stack_2.$$

规划片段等价的定义则稍微复杂,我们需要先明确关系的传递闭包.

$\mapsto$表示$\prec$的传递闭包,亦即:

如果$\alpha_i\mapsto\alpha_j$ 成立,则必须满足:(i) $\alpha_i\prec\alpha_j$;或(ii) 存在动作片段α_k,使得$\alpha_i\mapsto\alpha_k\wedge\alpha_k\prec\alpha_j$.

定义 8(规划片段等价). 给定两规划片段$\pi=\langle\Phi,R_\alpha\rangle$, $\pi'=\langle\Phi',R'_\alpha\rangle$,称π包含于π'(记为$\pi\subseteq\pi'$)当且仅当:

(i) π和π'是同一子问题的解;

(ii) $\forall r_{ij}\in R_\alpha$,必有 $r_{ij}\in R'_\alpha$,或 $\alpha_i\leqslant\alpha_j\in R'_\alpha$.

如果$\pi\subseteq\pi'$且$\pi'\subseteq\pi$,则$\pi=\pi'$.

理想状态下,如果规划片段与子问题之间存在一一对应关系,则我们可根据子问题唯一地还原出相应的规划片段解.但一般情况下,子问题与规划片段解之间并不存在一一对应关系,同一个子问题可能存在多个规划片段解.例如图 4 中的问题,对象 B 和 C 具有相同的子问题,但它们具有不同的规划片段解.

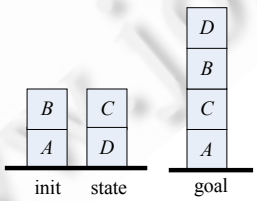


Fig.4 Sub-Problem of block B equals to that of block C, but the plan fragment solutions of them are different

图 4 积木块 B 和 C 具有相同的子问题,但它们的规划片段解是不同的

我们把规划片段$\pi\subseteq\pi'$和$\pi=\pi'$统称为相容关系;把π与π'不定时称为不相容关系,记为$\pi\perp\pi'$.

类似地,我们可以给出规划解相似的定义.

定义 9(规划片段的相似度). 给定两规划片段$\pi=\langle\Phi,R_\alpha\rangle$, $\pi'=\langle\Phi',R'_\alpha\rangle$,若$\pi=\pi'$,或$\pi\subseteq\pi'$,则 $Sim(\pi,\pi')=\dfrac{|\Phi|}{|\Phi'|}$表

示π和π'的相似度.

定义 10(规划解的相似度). 给定规划解 *Sol*,*Sol*′,若两者共有 k 个规划片段$\pi_1,\pi_2,\ldots,\pi_k$和 $\pi'_1,\pi'_2,...,\pi'_k$ 分别对应相似,则 $Sim(Sol,Sol')=\frac{\sum_{i=1}^{k}Sim(\pi_i,\pi'_i)}{k}$ 表示规划解 *Sol*,*Sol*′的相似度.

定义 11(解 *Sol* 的结构). 任意规划解 *Sol*,总可以表示成 $o_1\pi_1+o_2\pi_2+\ldots+(o_i+o_j)\pi_i+\ldots+o_n\pi_n$ 形式的规划片段的组合,其中,π_i 表示规划片段解.项$(o_i+o_j)\pi_i$ 表示对象 o_i,o_j 具有相同的规划片段.这些互异的规划片段集$\pi_1,\pi_2,\ldots,\pi_i,\ldots,\pi_n$构成了 *Sol* 的结构性知识,用符号$\Theta_{Sol}=\{\pi_1,\pi_2,\ldots,\pi_i,\ldots,\pi_n\}$表示.

后文的定理 3 和定理 4 给出了规划问题结构与规划解结构之间的关系.

定义 12(先验知识). 对于规划问题 *Prob*,*o* 是 *Prob* 中的规划对象,*Sol* 是 *Prob* 的一个解,对象 *o* 在解 *Sol* 下的先验知识$\mu(o)=\langle sp(o),\pi(o)\rangle$,或$\mu=\langle sp,\pi\rangle$,其中,$\pi$为从 *Sol* 中抽取的关于 *sp* 的规划片段解.

μ为类型 *t* 的先验知识,ψ表示先验知识集合.

例如,图 1 中问题被求解后,将产生$\mu_{block}=\langle block,A,\langle ontable_1,clear_1\rangle,pickup_1\prec stack_1\prec stack_2,\langle on_1,on_2\rangle\rangle$形式的先验知识,简记为$\mu_{block}=\langle ontable_1,clear_1\rangle pickup_1\prec stack_1\prec stack_2\langle on_1,on_2\rangle$.

3 SOLP 算法

本节给出 SOLP(算法 1)的详细描述.SOLP 可分成两个阶段:

- 第 1 阶段,SOLP 工作在学习状态,此时,SOLP 的输入之一 *Sol* 非空.SOLP 通过分析规划问题,识别单一对象的初始性质和目标性质,以此构成子问题.接着,SOLP 分析规划(解)以获取能使初始子状态转换成目标子状态的转换链.先验知识就由子问题和其转换链构成,并作为学习结果的输出被保存在领域文件里(先验知识库 PDB).
- 第 2 阶段,SOLP 工作在规划状态,此时,SOLP 的输入之一 *Sol* 为空.SOLP 在先验知识的帮助下搜索规划(解).这一阶段,SOLP 首先根据当前问题的子问题从 PDB 检索中先验知识,然后将检索到的先验知识实例化并合并成可满足问题的句子集,连同规划问题编码得到的句子集一并作为输入,运用 SAT 求解器进行求解,最后将求解结果转化成最终解.

算法 1 各个部分的详细描述将在第 3.1 节~第 3.3 节给出.

算法 1. Overview of SOLP.

Require: (1) A planning problem *Prob*, including planning domain *D*, initial state *I* and goal state *G*;
(2) A plan *Sol*, this input is empty when algorithm working in solving state;
(3) A list of prior knowledge ψ.

Ensure: A list of prior knowledge ψ or plan *Sol*.

1. **if** Sol is not empty **then**
2. *learning*(*Prob*,*Sol*,ψ)
3. **else**
4. *Sol*=*do_plan*(*Prob*,ψ)
5. **end if**
6. end.

接下来,我们将给出算法 1 子例程 *learning*(*Prob*,*Sol*,ψ)和 *Sol*=*do_plan*(*Prob*,ψ)的描述.

3.1 步骤1:构造子问题

子例程 Identify(算法 2)根据 *Prob* 构造子问题(对应图 3 中的"1?"),为此需要根据规划对象分别考虑初始状态和目标状态.对 *Os* 中每一规划对象 *o*,分别检查并识别 *o* 在初始状态拥有的性质 I_ρ和目标状态拥有的性质 G_ρ(步骤 2 和步骤 3),并表示成 $sp(o)=\langle t,o,I_\rho,G_\rho\rangle$.

算法 2. Identify sub-problems of *Prob*.

Require: (1) A planning problem *Prob*;

(2) A domain objects *o*.

Ensure: The sub-problems of *Prob*, *sp*.

1. Initialize $sp=\varnothing$;
2. identify the bag of initial properties of *o* typed *t*, I_ρ;
3. identify the bag of goal properties of *o* typed *t*, G_ρ;
4. build the sub-sproblem $sp(o)=\langle t,o,I_\rho,G_\rho\rangle$;
5. **return** *sp*;
6. end.

子问题将被用来组织和检索先验知识,这将分别在第 3.2 节和第 3.3 节介绍.

3.2 步骤2:学习先验知识

前一步获得了子问题 $sp(o)=\langle t,o,I_\rho,G_\rho\rangle$.先验知识的学习(例程 Learning)分别需要规划(解)*Sol*、子问题集 *SPs* 和当前先验知识列表这 3 个参数作为输入.这一过程对应图 3 中的"2?"部分.

SOLP 算法有两个不同的工作状态——学习状态和规划状态,这通过 SOLP 的 4 个输入参数中的第 3 个输入规划(解)*Sol* 是否为空来判断:如果输入 *Sol* 非空,则 SOLP 开始根据规划对象分析 *Sol*,以从中提取规划片段.这一过程首先识别规划对象 *o* 所对应的子问题 *sp*(步骤 2、步骤 3),然后从解 *S* 中抽取 *o* 的规划片段 $\pi(o)$(步骤 4),从而构造先验知识 μ_{new}(步骤 5),接着检查先验知识列表里 ψ 中是否已经有相等的先验知识(步骤 6),如果不存在相等的先验知识,则将 μ_{new} 添加到 ψ 中(步骤 7).

请注意,构造的 $\mu_{new}=\langle t,o,I_\rho,\pi,G_\rho\rangle$ 是与具体对象 *o* 无关的.换句话说,学习得到的先验库 PDB 与具体的规划对象无关,PDB 中的每条先验知识只记录了该条先验知识所涉及的对象类型等相关信息.

至此,规划片段已被抽取,并形成先验知识保存在先验知识库 PDB 中,这些先验知识库将被记录在领域描述文件中,对应图 3 中的"4?"部分.

算法 3. Learning prior knowledge.

Require: (1) A planning problem *Prob*;

(2) A plan *Sol*;

(3) A list of prior knowledge ψ.

Ensure: A list of prior knowledge ψ.

1. **for** each plan objects *o* in *Prob* **do**
2. identify the bag of initial properties of o_i, $I_\rho^{o_i}$;
3. identify the bag of goal properties of o_i, $G_\rho^{o_i}$;
4. extract $\pi(o)$ from *Sol*;
5. build prior knowledge $\mu_t=\langle t,\wedge,I_\rho,\pi,G_\rho\rangle$;
6. **if** no $\mu_t'\in\psi,\mu_t\in\mu_t'$ **then**
7. add μ_t into the list of prior knowledge ψ;
8. **end if**
9. **end for**
10. **end**.

3.3 步骤3:带先验知识的规划方法

这一步首先需要根据当前问题构造子问题(步骤 3),并根据子问题从先验知识库 PDB 中检索先验知识 ψ_r(步骤 4、步骤 5)(对应图 3 中的"3?"),然后将 ψ_r 例化成部分规划形式的先验知识 ψ_r'(步骤 7),最后,ψ_r' 被合并编

码成句子形式的先验知识 C_μ.得到 C_μ后,SOLP 将 C_μ连同问题编码得到的句子集 C_{Prob}(步骤 9)一并提交给 SAT 求解器进行求解(步骤 10).如果 SAT 求解器无法找到解,则增加 K,重复步骤 8~步骤 10;反之,SOLP 将成功终止并返回规划解.

关于 SAT 求解器如何寻找模型的算法,并非本文关注的内容,限于篇幅,本文将不作讨论.下面将对算法 4 中先验知识的例化、先验知识的合并、编码以及规划问题的编码这几部分进行详细解释.

算法 4. Planning with prior knowledge, *do_plan*(*Prob*,ψ).

Require: (1) A planning problem *Prob*;

(2) A set of prior knowledge ψ.

Ensure: A plan *Sol*.

1. $\psi_r \leftarrow \varnothing$;
2. **for** each plan objects $o \in O$ **do**
3. $sp(o)=identify(Prob,o)$;
4. retrieve prior knowledge $\mu(o)$ of $sp(o)$ from ψ;
5. put the prior knowledge μ in ψ_r;
6. **end for**;
7. Instantiate the prior knowledge ψ_r as ψ_a, $\langle \psi_a,k\rangle=Instantiate(I,\psi_r,G)$;
8. Merge ψ_a as a set of clause, $C_\mu=Merge(\psi_a,k)$;
9. Encode the planning problem *Prob* as a set of clause C_{Prob}, $C_{Prob}=Encode(Prob,k)$;
10. Solve the satisfiability problem using SAT Solver and return final plan $Sol=Solve(C_\mu \cup C_{Prob})$;
11. if no solution is found,we increment k by one, and go to Step 8;
12. **end**.

首先是算法 4 中先验知识的例化部分,为了解释 *Instantiate*(I,ψ_r,G)(算法 5),需要以下相关定义:

定义 13(部分例化的规划片段). 带参数的规划片段是二元组$\pi_x(o)=\langle M_x(o),R_m(o)\rangle$,其中, $M_x(o)=\{m_{i1}^o,...,m_{ik}^o\}$ 为部分例化动作模型集合, m_{ik}^o 表示动作模型 m_i 的第 k 个参数已被对象 o 例化的部分例化动作模型.$R_m(o)$为定义在 $M_x(o)$上的形如 $m_{ik}^o \prec m_{jk}^o$ 的约束集.

定义 14(实例化规划片段). 实例化规划片段是二元组$\pi_\alpha=\langle A(o),R_\alpha(o)\rangle$,其中, $A(o)=\{\alpha_{i1}^o,...,\alpha_{ik}^o\}$ 为基动作集, α_{ik}^o 表示第 k 个参数为 o 的基动作.$R_\alpha(o)$为定义在 $A(o)$上的形如 $\alpha_{ik}^o \prec \alpha_{jk}^o$ 的约束集.

类似地,我们可以给出部分例化先验知识以及实例化先验知识的定义如下:

定义 15(部分例化的先验知识). 对于规划问题 *Prob*,o 是 *Prob* 中一规划对象,*Sol* 是 *Prob* 的一个解,对象 o 在 *Sol* 下部分例化的先验知识是一个六元组$\mu_x(o)=\langle t,o,B,I_\rho,\pi_x,G_\rho\rangle$:

- t:对象 o 的类型;
- o:问题 *Prob* 中的规划对象;
- B:形如 $x=c$,$x=y$,或 $x \in D_x$ 的绑定约束集,D_x 为变量 x 的值域的子集;
- I_ρ:部分例化初始子状态;
- π_x:部分例化的规划片段;
- G_ρ:部分例化目标子状态.

有时,在不引起混淆的前提下,μ_x 也可表示成形如$\mu_x=\langle sp,\pi_x\rangle$的二元组形式,约定部分例化的先验知识集合用$\psi_x$表示.

定义 16(实例化的先验知识). 对于规划问题 *Prob*,o 是 *Prob* 中一规划对象,*Sol* 是 *Prob* 的一个解,对象 o 在 *Sol* 下实例化的先验知识是一个六元组$\mu_a(o)=\langle t,o,B,I_p,\pi_a,G_p\rangle$:

- t:对象 o 的类型;

- o:问题 *Prob* 中的规划对象;
- B:形如 $x=c$ 的绑定约束集;
- I_p:初始命题子状态;
- π_a:部分例化的规划片段;
- G_p:目标命题子状态.

在不引起混淆的前提下,μ_a 也可以表示成形如$\mu_a=\langle sp_p,\pi_a\rangle$的二元组形式,约定实例化的先验知识集合用$\psi_a$表示.

Instantiate 例程主要是执行先验知识的例化任务.我们将先验知识的例化过程总结成以下两个函数的形式:

定义 17(μ到μ_x的映射函数 f_x^o). 函数 $f_x^o:\psi\to\psi_x$ 各项之间的映射关系如下:

- $\pi\to\pi_x$:对每一$\alpha^i\in\Phi$,添加 m_i^o 到Φ_x中,其中,α是动作名为 m 的动作片段,添加绑定约束 $x=o$(变量 x 为动作模型 m 的第 i 个参数)到 B 中.对每一$\alpha\prec\beta\in R$,添加 $m_i^o\prec n_j^o$ 到 R_x,其中,α与 m 具有相同的动作名称,β与 n 有相同的动作名称.
- $I_\rho\to I_\zeta$:对 I_ρ中每一性质ρ^λ,如果 M_x 中存在一个 m_λ^o,m_λ^o 存在一个前提条件: ζ_λ^o 与ρ^λ具有相同的谓词名称,则将ρ^λ映射成 ζ_λ^o.
- $G_\rho\to G_\zeta$:对 G_ρ中每一性质ρ^γ,如果 M_x 中存在一个 m_γ^o,m_γ^o 存在一个前提条件: ζ_γ^o 与ρ^γ具有相同的谓词名称,则将ρ^γ映射成 ζ_γ^o.

定义 18(μ_x 到μ_α的映射函数 $f_\alpha^{\langle I,G\rangle}$). 函数 $f_\alpha^{\langle I,G\rangle}:\psi_x\to\psi_\alpha$:各项之间的映射关系如下:

- 对$\pi_x=\{M_x,R_m\}$,根据性质 2 确定形如 $x=y$ 的绑定约束,将这些约束添加到 B 中.
- $I_\zeta\to I_p$:对 I_ζ中每一 ζ_λ^o, 如果 I 中存在命题 p,p 是由 ζ_λ^o 进一步例化得到的,则将 ζ_λ^o 映射为 p,并根据 p 绑定 ζ_λ^o 中其他变量,将得到的绑定约束添加到 B 中.
- $G_\zeta\to G_p$:对 G_ζ中每一 ζ_γ^o, 如果 G 中存在命题 g,g 是由 ζ_γ^o 进一步例化得到的,则将 ζ_γ^o 映射为 g,并根据 g 绑定 ζ_γ^o 中其他变量,将得到的绑定约束添加到 B 中.
- $\pi_x\to\pi_a$:对 M_x 中每一 m_i,如果初始状态中存在一命题 p 可通过 m_i 中某一前提条件ζ_λ例化得到,则将ζ_λ映射为 p,并根据 p 绑定ζ_λ中其他变量,将得到的绑定约束添加到 B 中;对 M_x 中每一 m_i,如果目标状态中存在一命题 q 可通过 m_i 中某一添加效果ζ_γ例化得到,则将ζ_γ映射为 q,并根据 q 绑定ζ_γ中其他变量,将得到的绑定约束添加到 B 中;最后,根据绑定约束 B,将 M_x 中的每一 m_i 映射为 a_i,将 $r_m=(m_i\prec m_j)$映射成 $r_a=(a_i\prec a_j)$.

算法 5. Instantiate the prior knowledge, *Instantiate*(I,ψ_r,G).

Require: (1) Initial state I;
(2) Goal G;
(3) The prior knowledge ψ_r.

Ensure: Prior Knowledge ψ_a, action number K.

1. $A_{total}\leftarrow\varnothing$, $\psi_a\leftarrow\varnothing$;
2. **for** each $\mu=\langle sp,\pi\rangle\in\psi_r$ **do**
3. partially instantiate μ according to function f_x^o, $\mu_x=f_x^o(\mu)$;
4. fully instantiate according to function $f_a^{\langle I,G\rangle}$, $\mu_a=f_a^{\langle I,G\rangle}(\mu_x)$; ;
5. $A_{total}\leftarrow A_{total}\cup A$;
6. put the μ_a into ψ_a;
7. **end for**;
8. **return** ψ_a, $K=|A_{total}|$;

9. **end**.

算法 5 首先利用规划问题中动作模型的定义,结合具体对象信息,将先验知识μ部分例化为先验知识μ_x(步骤 3);然后,利用规划问题初始状态 I 和目标状态 G,将μ_x进一步例化成μ_a(步骤 4);最后,返回实例化先验知识集ψ_a以及先验知识中包含的动作个数 K(步骤 8).

这一过程的第 2 部分涉及到将例化后的先验知识ψ_a合并编码成可满足的句子,合并的过程要求在保持各先验知识中动作序关系的前提下将先验知识$\mu_a\in\psi_a$合并为全局先验知识.本文的合并逻辑与 Mali[16]的类似,合并后的全局先验知识μ'必须包含被合并的先验知识μ_1,μ_2中的所有约束.但我们的工作与 Mali 的不同之处有以下几点:(1) 要合并的先验知识中可能存在相同的动作,即,μ_i的第 j 个动作 a_{ij} 可能与$\mu_{i'}$的第 j'个动作 $a_{i'j'}$相同($a_{ij}=a_{i'j'}$)的情况;(2) 将ψ_a中的动作合并到 K 个时间步上,形成长度为 K 的部分规划,而如果按照 Mali 的做法,是将要合并的子规划按任意顺序排列,然后将这一个排列复制 K 个拷贝,得到长度为 K^2 的序列;(3) 我们的合并逻辑不允许合并过程中删除动作;(4) 暂不考虑先验知识中动作之间的因果链约束(先验知识中,动作之间的因果链约束显然是一类可被进一步利用的重要知识,这部分工作将在本文后续工作中进一步开展).根据以上几点,我们的合并逻辑可表示成以下几类约束:

(C_1): $\wedge_{q=1}^{K}((\vee_{i=1}^{m}\vee_{j=1}^{k_i}a_{ij}(q))\vee\varnothing(q))$, 表明每一时间步 q 要么是空动作,要么是ψ_a中至少 1 个动作.

(C_2): $\wedge_{i=1}^{m}\wedge_{j=1}^{k_i}(\vee_{q=0}^{K}a_{ij}(q))$, 表明$\psi_a$中的每一动作至少被分配到 1 到 K 的某个时间步.

(C_3): $\wedge_{q=1}^{K}(\wedge_{i=1}^{m}\wedge_{j=1}^{k_i}(q<j\Rightarrow\neg a_{ij}(q)))$, 表明当 $q<j$ 时,ψ_a中的第 i 条先验知识的第 j 个动作不可能被分配到 0 到 j−1 时间步,以确保第 i 条先验知识的第 j 个动作之前的动作能够被分配到 0 到 q−1 时间步之间.

(C_4): $\wedge_{q=1}^{K}(\wedge_{i=1}^{m}\wedge_{j=1}^{k_i}(k-q<k_i-j\Rightarrow\neg a_{ij}(q)))$, 要求当$(K-q)<(k_i-j)$时,在$\psi_a$中的第 i 条先验知识的第 j 个动作不能被分配到 j+1 到 K 时间步,以确保第 i 条先验知识的第 j 个动作之后的动作能被分配到 q+1 到 K 时间步之间.

(C_5): $\wedge_{q=1}^{K}(\wedge_{i=1}^{m}\wedge_{j=1}^{k_i}(a_{ij}(q)\Rightarrow\neg\phi(q)))$, 表明实动作 a_{ij} 和空动作ϕ不能同时分配到同一时间步 q 上.

(C_6): $\wedge_{q=1}^{K}(\wedge_{i=1}^{m}\wedge_{j=1}^{k_i}(a_{ij}(q)\Rightarrow\wedge_{q_1\neq q}\neg a_{ij}(q_1)))$, 表明动作 a_{ij} 在时间步 q 上成立,则不能在其他时间步 $q_1(q_1\neq q)$上成立.

(C_7): $\wedge_{q=1}^{K}(\wedge_{i=1}^{m}\wedge_{j=1}^{k_i}(a_{ij}(q)\Rightarrow\wedge_{q_1=1}^{q-1}\wedge_{j_1=j+1}^{k_i}(\neg a_{ij_1}(q_1))))$, 表明动作 a_{ij} 在时间步 q 上成立,则 a_{ij} 之后的动作不能被分配到时间步 0 到 q−1 上.

(C_8): $\wedge_{q=1}^{K}(\wedge_{i=1}^{m}\wedge_{j=1}^{k_i}(a_{ij}(q)\Rightarrow\wedge_{q_1=q+1}^{K}\wedge_{j_1=1}^{j}(\neg a_{ij_1}(q_1))))$, 表明动作 a_{ij} 在时间步 q 上成立,则 a_{ij} 之前的动作不能被分配到时间步 q+1 上.

(C_9):$\forall a_{ij}=a_{i'j'}$, $\wedge_{q=1}^{K}(a_{ij}(q)\Rightarrow\wedge_{q_1\neq q,q_1=1}\neg a_{i'j'}(q_1))$, 表明若不同先验知识之间存在相同动作,即 $a_{ij}=a_{i'j'}$,当 a_{ij} 在时间步 q 成立,则 $a_{i'j'}$不能在其他时间步上成立.

(C_{10}):$\forall a_{ij}=a_{i'j'}$, $\wedge_{q=1}^{K}(a_{ij}(q)\Rightarrow\wedge_{q_1=1}^{q}\wedge_{j_1=j'+1}^{k_i'}\neg a_{i'j_1}(q_1))$, 表明若不同先验知识之间存在相同动作,即 $a_{ij}=a_{i'j'}$,则 a_{ij} 在时间步 q 成立,则 $a_{i'j'}$之后的动作不能在时间步 q 之前成立.

(C_{11}):$\forall a_{ij}=a_{i'j'}$, $\wedge_{q=1}^{K}(a_{ij}(q)\Rightarrow\wedge_{q_1=q+1}^{K}\wedge_{j_1=0}^{j'}\neg a_{i'j_1}(q_1))$, 表明若不同先验知识之间存在相同动作,即 $a_{ij}=a_{i'j'}$,则 a_{ij} 在时间步 q 成立,则 $a_{i'j'}$之前的动作不能在时间步 q 之后成立.

这一过程的第 3 部分涉及到将规划问题编码成可满足句子,该项工作起源于 Kautz 等人[17]的工作,有大量文献介绍如何将规划问题编码成可满足问题[17−21].基于内容的完整性考虑,本文介绍 SOLP 系统中使用的规划问题编码方法,该方法与 SatPlan2006 中使用的编码方法完全相同.

具体来说,对于步长为 K 的有界规划问题,我们按照以下方式进行编码:

(C_{12}): $\wedge_{u_i\in I}u_i(0)$, 表明初始状态中的命题将在时间步 0 成立.

(C_{13}): $\wedge_{u_i\in G}u_i(K)$, 表明目标状态的命题将在时间步 K 成立.

(C_{14}): $\wedge_{a_i\in A}\wedge_{t=1}^{K}(a_i(t)\Rightarrow\wedge_{u_j\in Pre(a_i)}u_j(t))$, 其中,$A$ 表示所有的实例动作集,$Pre(a_i)$表示动作 a_i 的前提条件集合.

该约束表明:对 A 中的任一动作 a_i,若该动作在时间步 t 执行($a_i(t)$),则该动作的所有前提条件 u_j 必须在时间步 t 成立($u_j(t)$).

(C_{15}): $\wedge_{u_i\in U}\wedge_{t=1}^{K}(u_i(t)\Rightarrow\vee_{u_i\in Add(a_j)}a_j(t-1))$, 其中,$U$ 表示所有的命题集,$Add(a_j)$表示动作 a_j 的添加效果集.该约束表明:对 U 中每一命题 u_i,若该命题在时间步 t 成立($u_i(t)$),则至少存在 1 个以 u_i 为添加效果的动作 a_j 在前一时间步 t–1 成立($a_j(t-1)$).

(C_{16}): $\wedge_{u_i\in U}\wedge_{t=1}^{K}(u_i(t)\Rightarrow\vee_{(u_i,u_j)\in Mutex\_P(t)}\neg u_j(t))$, 其中,$Mutex_P(t)$表示在时间步 t 上互斥[**]的两两命题集.该约束表明:若 $u_i(t)$在时间步 t 成立,则所有与 u_i 在时间步 t 互斥的命题 u_j 均不可能成立($\neg u_j(t)$).

(C_{17}): $\wedge_{a_i\in A}\wedge_{t=1}^{K}(a_i(t)\Rightarrow\wedge_{(a_i,a_j)\in Mutex\_A(t)}\neg a_j(t))$, 其中,$Mutex_A(t)$表示时间步 t 上互斥的两两动作集.该约束表明:若 a_i 在时间步 t 成立($a_i(t)$),则所有与 a_i 在时间步 t 互斥的动作 a_j 均不可能成立($\neg a_j(t)$).

根据约束 C_{12}~约束 C_{17},我们即可将有界规划问题编码成可满足问题.最终,我们将分别得到表示先验知识的句子集合 C_μ和表示要求解的规划问题的句子集合 C_{Prob}.此时,作为这个过程的最后一个步骤,需要将 C_μ和 C_{Prob}一并提交给 SAT 求解器进行求解.由于 SOLP 系统中问题编码和求解是分模块进行的,编码和求解之间没有耦合关系,因此可以很方便地扩展使用各种不同的 SAT 求解器.这一点与 SatPlan2006 系统类似.

算法 6. Merge ψ_a as a set of clause, $Merge(\psi_a,K)$.

Require: (1) A set of instantiated prior knowledge ψ_a;

(2) A time step K.

Ensure: A set of clause C_μ.

1. $C_\mu\leftarrow\varnothing$;
2. Merge and Encode the instantiated prior knowledge ψ_a as satisfiability according to constraints C_1~C_{11} and the argument K;
3. put the result clause into C_μ;
4. return C_μ;
5. **end**.

4 SOLP 的性质

规划领域的研究者们很早就意识到:如果规划器能够重用部分旧规划解或规划解中的某些知识,避免总是从零构造规划,可能是提高规划求解效率的一种方法.然而,Nebel 等人[13]的结果告诉我们:理论上,规划重用并不能带来效率的提高,甚至规划重用可能比规划求解问题更困难.规划重用往往涉及到规划问题匹配的问题,即检索与当前规划问题最相似的问题,通过提取该相似问题的解,最后对检索到的解进行最少修改、调整从而得到当前问题的解.极端情况下,如果两问题 $Prob$ 和 $Prob'$其初始状态和目标状态中所有命题均匹配,则可通过对 $Prob$ 的解执行变量替换,即可得到 $Prob'$的解.然而,Nebel 等人[13]的结果告诉我们:纵使在初始状态为空的前提下,不同规划问题之间的最优匹配问题仍然是 NP-hard 的.

SOLP 的特点在于,不需要采用类似 Nebel 等人使用的命题匹配的方法来检索/识别相似问题.下面的定理告诉我们,本文使用的用对象子状态的匹配来衡量状态之间的相似程度的办法与 Nebel 使用的方法两者具有本质上的不同.

定理 1. 给定两状态 s_1,s_2,其中,状态 s_1 涉及 m 个规划对象 $o_1,\ldots,o_m\in O$,状态 s_2 涉及 n 个规划对象 $o'_1,\ldots,o'_n\in O$. 假定对象 $o_i(i=1,\ldots,m)$在状态 s_1 下的对象子状态 ss_i 与 o'_j $(j=1,\ldots,n)$在状态 s_2 下的子状态 ss_j 相等,称 s_1 下对象 o_i 与 s_2 下的对象 o'_j 的子状态匹配.假定有从 s_1 到 s_2 的映射函数 $\eta:O\rightarrow O'$,使 s_1 与 s_2 间有最大命题匹配,则 η不一定使 s_1 与 s_2 间有最大对象子状态匹配;反之亦然.

** 互斥的概念最初由 Blum 提出,其基于规划图的规划技术[22]在智能规划领域中被广泛使用.

证明:先证明在映射η下使 s_1 与 s_2 有最大命题匹配时,s_1 与 s_2 间不一定有最大对象子状态匹配.这只需要给出一个反例即可证明.考察状态:

- s_1={*ontable*(*E*),*on*(*D*,*E*),*on*(*C*,*D*),*on*(*B*,*C*),*on*(*A*,*B*),*clear*(*A*)};
- s_2={*ontable*(*F*′),*on*(*E*′,*F*′),*on*(*D*′,*E*′),*on*(*C*′,*D*′),*on*(*B*′,*C*′),*on*(*A*′,*B*′),*clear*(*A*′)}.

令η:$A\to A'$,$B\to B'$,$C\to C'$,$D\to D'$,$E\to E'$,此时,$\eta(s_1)\cap s_2$=5 为最大命题匹配.但在η下,s_1 与 s_2 并非最大对象子状态匹配.这表明,最大命题匹配不能保证一定是最大对象子状态匹配.

下面证明使 s_1 与 s_2 有最大对象子状态匹配的映射η'也不能保证一定是最大命题匹配.

令η':$A\to A'$,$B\to D'$,$C\to E'$,$D\to B'$,$E\to F'$,此时,s_1 与 s_2 中分别有 5 个对象的子状态相等.因此,η'下,s_1 与 s_2 最大对象子状态匹配,但此时,$\eta'(s_1)\cap s_2$=3,并非最大命题匹配. □

上述定理表明,最大命题匹配与最大对象子状态匹配两者本质上是不一样的.这也说明,Nebel 等人[13]的结论并不适合我们的方法.

由于对象子状态只与领域模型有关,因此,我们很容易得到以下引理:

引理 1. 给定规划领域 *D*,假定该领域描述需要用到 *m* 个不同谓词,这 *m* 个不同谓词所含参数个数分别为 $n_1,n_2,\ldots,n_m$,n=max($n_1,n_2,\ldots,n_m$),显然,$\Omega_D\leqslant m\times n$,即,领域 *D* 所有性质集合$\Omega_D$的基数不超过 $m\times n$.

证明:显然,对于第 i(i=1,…,m)个谓词,可产生$\rho^1,\ldots,\rho^{n_i}$ 共 n_i 个性质,因此,$|\Phi_D|=n_1+\ldots+n_m\leqslant m\times n$. □

类似地,我们可以得到另一个引理:

引理 2. 给定含 l 个不同动作模型的领域 *D*,这 l 个不同动作模型所含参数个数分别为 $h_1,\ldots,h_l$,h=max($h_1,\ldots,h_l$),则$|\Phi_D|\leqslant l\times h$,即,动作片段集$\Phi_D$的基数不超过 $l\times h$.

证明:显然,对于第 j(j=1,…,l)个动作模型,可产生$\alpha^1,\ldots,\alpha^{h_j}$ 共 h_j 个动作片段,因此,$|\Phi_D|=h_1+\ldots+h_l\leqslant l\times h$. □

SOLP 的基本思想源于利用Ω_D中的性质描述状态及问题的结构知识,用Φ_D中的动作片段描述规划解的结构知识.

定理 2. 任一给定的规划领域 *D*,总存在一完备的先验知识库,记为ψ_{CD}.

证明:根据引理 1,可利用Ω_D中性质穷尽所有可能的组合,得到所有可能的合法对象子状态,记为 *SS*.又根据子问题的定义,*SS* 中任一子问题 *sp*,我们总能从引理 2 中的Φ_D选择适当的动作片段序列,使得该动作片段序列构成 *sp* 的解.因此,可将 *SP* 中的每一问题均通过Φ_D为之构造规划片段解,从而得到完备的先验知识库ψ_{CD}. □

定理 2 表明:随着 SOLP 求解次数的增长,SOLP 最终会达到所有问题都曾经遭遇过的所谓“经验丰富”的状态.

推论 1. 先验知识库(PDB)中的子问题是规划问题结构知识的并.

证明:由于任意规划问题 *Prob* 总可以由 *SP* 中子问题线性表示成 $o_1sp_1+o_2sp_2+\ldots+(o_i+o_j)sp_i+\ldots+o_nsp_n$ 的形式,对于具有相同子问题的对象 o_i,o_j,SOLP 只保留其中一个子问题对应得到的先验知识(算法 3).换言之,SOLP 学到的先验知识是由反映规划问题的结构知识的子问题以及该子问题所对应的规划片段所构成.从而,先验知识库(PDB)中的子问题是规划问题结构知识的并. □

定理 2 和推论 1 保证了先验知识库(PDB)的简洁性,而不致随着问题遭遇越多而过度膨胀.

定义 19(**问题结构等价**). 对于规划问题 *Prob*,*Prob*′,若$\forall sp\in\Theta_{Prob},\exists sp'\in\Theta_{Prob'}$,使得 *sp*=*sp*′,称 *Prob*,*Prob*′两问题结构等价,记为$\Theta_{Prob}=\Theta_{Prob'}$.

推论 2. SOLP 的学习只在问题结构等价或最相似的问题之间发生.

证明:SOLP 早期,由于遭遇的问题不多,因此其先验知识ψ比较少,没有达到完备的先验知识ψ_{CD}的状态,因此会导致当前要求解的问题 *Prob* 的某些子问题在现有的先验知识库ψ中没有相应的先验知识与之对应;或者 SOLP 遭遇过多同质化的问题,SOLP 所拥有的先验知识ψ比较单一,这同样会导致当前要求解的问题 *Prob* 的某些子问题在现有的先验知识库ψ中没有相应的先验知识与之对应.当这两种情况出现时,由算法 4 可知,SOLP 可保证学习是在两个最相似的问题之间发生.若对当前要求解的问题 *Prob* 的每一子问题,ψ中都有相应的先验知识与之对应,则 SOLP 的学习是在两个结构等价的问题之间发生(算法 4). □

推论 2 保证了 SOLP 的学习机制的有效性.而下面的讨论则确保学习到的知识的准确性.

关于规划片段π(定义 2)、性质ρ和子问题 sp(定义 1)之间的关系,我们有以下相关性质成立:

性质 1. 给定规划问题 $Prob$ 以及该问题的解 Sol,o 为问题 $Prob$ 中的一对象,I 是 $Prob$ 的初始状态,G 是目标状态, Sol_F^o 是子问题 $sp(o)$的解片段,即, Sol_F^o 能将 I_p^o 转换为 G_p^o.

证明:该性质可直接由规划片段(3)以及规划解的定义得到. □

性质 2. 给定规划解 Sol 的一个解片段$\pi= Sol_F^o$,相应的部分例化规划片段$\pi_x(o)=\langle M_x,R_m\rangle$,若 $P(x_1,\ldots,x_{\lambda-1},o,x_{\lambda+1},\ldots,x_n),P(y_1,\ldots,y_{\lambda-1},o,y_{\lambda+1},\ldots,y_n)$均是被 o 部分例化的谓词,$\forall m_i,m_j\in M_x$,如果 $r_m=(m_i\prec m_j)\in R_m,P(x_1,\ldots,x_{\lambda-1},o,x_{\lambda+1},\ldots,x_n)\in Add(m_i),P(y_1,\ldots,y_{\lambda-1},o,y_{\lambda+1},\ldots,y_n)\in Pre(m_j)$,则 $x_1=y_1,\ldots,x_{\lambda-1}=y_{\lambda-1},x_{\lambda+1}=y_{\lambda+1},\ldots,x_n=y_n$.

证明:假设 $P(x_1,\ldots,x_{\lambda-1},o,x_{\lambda+1},\ldots,x_n)\neq P(y_1,\ldots,y_{\lambda-1},o,y_{\lambda+1},\ldots,y_n)$,则总可选择满足 $P(x_1,\ldots,x_{\lambda-1},o,x_{\lambda+1},\ldots,x_n)\in Pre(m),P(y_1,\ldots,y_{\lambda-1},o,y_{\lambda+1},\ldots,y_n)\in Add(m)$的部分例化动作模型 m,将其并入 M_x,即 $M'_x=M_x\cup m$.将约束 $r_1=(m_i\prec m)$, $r_2=(m\prec m_j)$并入 R_m,并删除 $r=(m_i\prec m_j)$约束,即 $R'_m=R_m\cup r_1\cup r_2/r_m$,从而得到 $\pi'=\langle M'_x,R'_x\rangle$. 显然, $\pi'=Sol_F^o$, 这与 $\pi=Sol_F^o$ 矛盾.从而,$x_1=y_1,\ldots,x_{\lambda-1}=y_{\lambda-1},x_{\lambda+1}=y_{\lambda+1},\ldots,x_n=y_n$. □

性质 2 是定义 15 中的先验知识的例化的基础.

定义 20. 对于规划问题 $Prob$,$Prob'$,其对应的规划解分别为 Sol,Sol'.若$\forall\pi\in\Theta_{Sol},\exists\pi'\in\Theta_{Sol'}$,使得$\pi=\pi'$,则称 $Prob$,$Prob'$两问题的解结构等价,记为$\Theta_{Sol}=\Theta_{Sol'}$.

显然,结构等价是比相似更强的一个概念.对于规划问题的结构与解的结构的关系,我们有以下定理成立:

定理 3. 解结构等价是规划问题结构等价的充分条件.

证明:该定理的证明可由规划问题结构、规划解结构以及规划片段解的定义直接推出. □

但规划问题结构等价并不能保证相应的规划解的结构等价,这是因为对于同一子问题,其对应的规划片段存在除等价关系外的另外两种关系,即,$\pi\subseteq\pi'$,$\pi\perp\pi'$的情况.

然而,我们有以下定理保证 SOLP 通过子问题检索规划片段的机制是合理的:

定理 4. 给定规划问题 $Prob$,$Prob'$,其对应的规划解分别为 Sol,Sol',$Sim(Sol,Sol')$关于 $Sim(Prob,Prob')$单调不减.

证明:根据定义 6,两问题越相似,则 $Sim(Prob,Prob')$值越大,这表明有越多的子问题相等.而同一子问题对应的规划片段存在$\pi=\pi'$,$\pi\subseteq\pi'$,$\pi\perp\pi'$这 3 种情况,在前两种情况下,$Sim(Sol,Sol')$会升高;而第 3 种情况由于π,π'存在不同的动作,则无法保证 $Sim(Sol,Sol')$会升高. □

5 实验结果

本文的实验系统是在 Inspiron-N4050,Ubuntu Linux 2.6.35-27-generic 内核,Intel(R) Core(TM) i3-2350M, CPU@2.30GHz×4,1.9GB 内存环境下运行并测试得到的结果.

为了方便同行之间的交流,我们忠实地记录了实验过程中观察到的一些结果.这些结果主要有:RT 代表算法运行的实际总时间;UT 表示算法运行用户曾代码所耗费的时间,不包括系统核心调用所耗费的时间;ST 则代表算法运行过程中耗费在核心调用层的时间;CNF 表示编码所使用的 CNF 变量数;$Clause$ 表示编码得到的句子数.测试时,首先让 SOLP 处于学习状态,对各问题实例逐次进行求解;当成功找到解后,SOLP 将结合当前问题,从解中提取先验知识,这些先验知识主要按照“子问题-规划片段”的形式进行表示,并将被保存在各自的领域描述文件 domain.pddl 中.

为使 SOLP 能够从领域描述文件 domain.pddl 中正确读取和保存先验知识,我们在 SatPlan2006 的基础上重写了 lex-fct_pddl.l,lex-ops-prior_pddl.l 以及 scan-fct_pddl.y,scan-ops-prior_pddl.y 等相关文件读写函数.其中,前两者分别是规划问题和规划领域的词法分析文件,后两者分别是规划问题和规划领域的语法分析文件.由于 SOLP 每次成功找到解后均会自动更新相应的先验知识库,因此,实验时对每一测试问题让 SOLP 求解两次,记录 SOLP 在二次求解时的性能表现,然后用 SatPlan2006 对同样的问题进行求解,从而得到 SOLP 在有先验知识

情形下与 SatPlan2006 的性能对比(见表 1 和表 2).为了能够更直观地认识 SOLP 和 SatPlan2006 在性能上的差异,我们在表 1 基础上绘制了 SOLP 和 SatPlan2006 在求解 block 领域的时间性能对比曲线图(如图 5 所示).

Table 1 Performance comparison of SOLP and SatPlan2006 on block domain

表 1 SOLP 和 SatPlan2006 在 block 领域中的性能对比

Problem	Planner	*RT*	*UT*	*ST*	*CNF*	*Clause*
Sussman	SOLP	0.03	0	0	244	1 903
	SatPlan2006	0.05	0	0	231	1 152
12-step	SOLP	0.15	0.07	0.03	1 622	32 574
	SatPlan2006	0.16	0.07	0.02	1 581	25 892
Large-a	SOLP	0.25	0.2	0.03	3 604	122 157
	SatPlan2006	0.47	0.34	0.04	2 829	84 072
Large-b	SOLP	1.59	1.43	0.12	8 085	417 012
	SatPlan2006	2.33	2.05	0.18	6 949	322 688
Large-c	SOLP	18.3	17.49	0.42	22 412	2 059 828
	SatPlan2006	34.66	33.05	1.33	20 321	1 714 216
Large-d	SOLP	65.98	63.92	1.28	47 337	6 766 187
	SatPlan2006	327.78	364.84	6.41	44 063	5 960 599

Table 2 Performance comparison of SOLP and SatPlan2006 on logistics domain

表 2 SOLP 和 SatPlan2006 在 logistics 领域中的性能对比

Problem	Planner	*RT*	*UT*	*ST*	*CNF*	*Clause*
prob001-log-easy	SOLP	0.23	0.01	0.01	1 330	6 431
	SatPlan2006	0.21	0.03	0	1 297	6 062
prob002-rocket-a	SOLP	2.35	1.86	0.2	5 870	48 012
	SatPlan2006	0.47	0.08	0.02	1 547	9 426
prob003-rocket-a	SOLP	0.46	0.18	0.03	5 945	48 602
	SatPlan2006	0.54	0.07	0.02	1 623	10 108
prob004-log-a	SOLP	0.44	0.52	0.15	6 293	41 306
	SatPlan2006	0.52	0.15	0.02	2 822	14 239
prob005-log-b	SOLP	0.59	0.2	0.03	3 367	20 394
	SatPlan2006	0.84	0.25	0.03	3 314	18 841
prob006-log-c	SOLP	0.44	0.21	0	5 818	38 958
	SatPlan2006	0.92	0.39	0.05	4 238	24 947

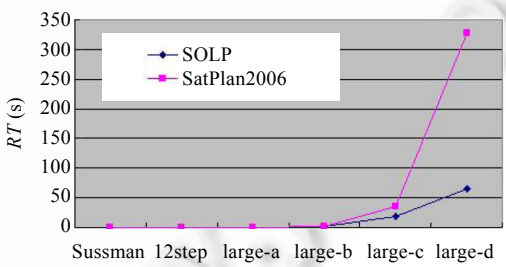


Fig.5 Time performance curve comparison of SOLP and SatPlan2006 on blocks domain

图 5 SOLP 和 SatPlan2006 在 block 领域的时间性能对比曲线图

从表 1 和图 5 来看:对于 bw-sussman 和 bw-12step 问题,SOLP 和 SatPlan2006 两者在时间性能表现上差异并不明显,SOLP 只比 SatPlan2006 略好;但对于 bw-large-a,bw-large-b,bw-large-c,bw-large-d 问题,SOLP 求解效率明显优于 SatPlan2006,尤其值得强调的是,SOLP 在求解 bw-large-d 问题时,只需要 65.98s 的时间即能返回正确解,比 SatPlan2006 需要 327.78s,提高了近 80%的效率.SOLP 求解效率的大幅提高主要得益于以下几点:

(1) 由于先验知识的存在,使得 SOLP 在求解前能够根据先验知识估计解的长度,而不是像 SatPlan2006 和大多数其他经典规划器那样盲目猜测规划解的长度,这避免了 SOLP 的过多无效搜索.

(2) 从状态空间的角度来看,先验知识中的动作片段排除了状态空间中过多的无效分枝.

(3) 从规划空间角度来看,规划片段提供了动作之间的部分序关系,排除了那些明显非法的动作序关系.

但 SOLP 所使用的 CNF 变量和句子数均比 SatPlan2006 多,这一点可从表 1 看出来.导致 SOLP 所使用的 CNF 变量和句子数均比 SatPlan2006 多的原因,主要是因为 SOLP 需要对先验知识进行编码.

类似地,在先验知识的指导下,我们用 SOLP 分别对 logistics 领域的基准测试例子进行求解,然后用 SatPlan2006 对这些问题进行求解,求解的结果记录在表 2 中.

同样地,为了能够更直观地认识 SOLP 和 SatPlan2006 在性能上的差异,我们在表 2 的基础上绘制了 SOLP 和 SatPlan2006 在求解 logistics 领域的时间性能对比曲线图(如图 6 所示).

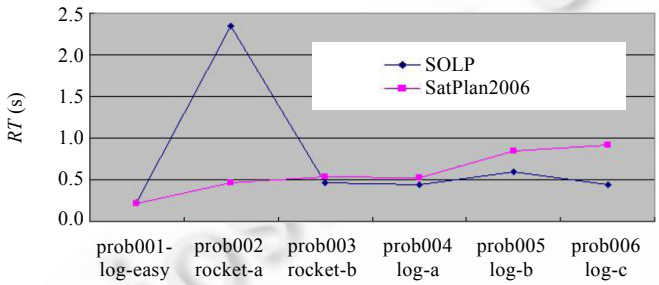


Fig.6 Time performance curve comparison of SOLP and SatPlan2006 on logistics domain

图 6 SOLP 和 SatPlan2006 在 logistics 领域的时间性能对比曲线图

从表 2 和图 6 来看,SOLP 在求解 logistics 领域时未能像在求解 block 领域时性能表现得那么突出,它只比 SatPlan2006 取得了相对的优势.究其原因,主要有以下几个方面:

(1) logistics 领域中的问题各个对象子状态高度重叠,这影响了学习先验知识的准确度以及其学习效率.

(2) logistics 领域的问题其对应的解往往高度并行,同一时间步可以执行多个动作,这导致 SOLP 根据学习到的先验知识估计解的长度准确性下降.

6 相关工作

规划领域的研究者们很早就意识到领域相关的控制规则可极大地压缩搜索空间,可为规划求解带来几个数量级的效率的提高[23−25].但准确、有效的领域知识的获取是一件困难且耗的时工作,这促使规划领域的研究者们尝试用机器学习的方法自动学习领域相关知识,以期提高规划求解效率.为了促进这方面的研究,国际智能规划大赛自 IPC-6(2008)开始,专门开辟了针对规划中学习问题研究的竞赛轨道 learning track.

从国际智能规划大赛来看,根据学习的知识类型以及使用的知识方法的不同,基于学习的规划系统大致可分成用于指导搜索的策略的学习、宏动作的学习、用于指导搜索的启发式数值函数的学习以及规划器组合配置的学习这 5 类不同的系统.PbP.s 系统[12]在 IPC-6 学习轨道的比赛中获得总体表现最优奖,但 PbP.s 的效率并没有令人信服地超越非学习的规划器,尤其是 PbP.s 系统在加学习机制和不加学习机制下性能表现上相差居然不大,因此,与其说 PbP.s 的学习机制提高了它的求解效率,倒不如说是它用了更好的非学习的规划器.与 PbP.s 系统相比,另一个系统 ObtuseWedge[4]则表现出更佳的学习效果,ObtuseWedge 系统在使用学习机制和不使用学习机制时性能表现有明显的差异.因此,IPC-6 将最佳学习奖颁给了 ObtuseWedge 系统.IPC-7(2011)的 learning track 比赛中共有包括 PbP2.quality 系统在内的 8 个规划器参加了比赛,最终,PbP2.quality 系统获胜.遗憾的是,IPC-6 中学习机理更清楚的 ObtuseWedge 系统因未知原因而未能在 IPC-7(2011)的 learning track 比赛中出现.

然而,学习轨道的比赛结果表明,学习机制的使用并没有使带学习能力的规划器在效率上彻底征服传统的不带学习能力的规划器.这一结果似乎证实了文中的“学习不能从理论上提高规划求解效率的结论”.然而正如本文早前的论述,Nebel 等人[13]的结论是基于命题匹配来衡量问题之间相似程度这一前提得到的.本文工作是基于对象子状态匹配来衡量问题之间相似程度的,因此,Nebel 等人的结论并不适用于本文的 SOLP 系统.理论分

析和实验结果支持了本文的学习能够提高规划求解效率的结论.

我们的工作中使用了性质作为刻画对象子状态和规划问题结构知识的工具.最早使用性质这一概念的是Fox 等人[14],他们在性质的基础上定义了一种性质相关结构(property relating structure),并利用动作模型构造性质转换规则,然后根据这些转换规则构造规划领域的类型结构(type structure),最终实现状态不变量(state invariants)的自动提取.动作片段、规划片段则是本文提出的全新概念.用对象子状态刻画问题的结构知识,用规划片段刻画规划解的结构知识,从结构的角度给出先验知识的表示、学习、使用是本文的核心工作,这也是本文的首创.

De la Rosa 等人的工作[26]着眼于学习能被使用于 TLPlan 的领域控制规则.他们的方法使用了包括复合谓词(compound predicates)、抽象谓词(abstracted predicates)和递归谓词(recursive predicates)的派生谓词来描述复杂的领域控制规则,其中的抽象谓词(abstracted predicates:谓词 $on(A,B)$可产生抽象谓词 abs_on_A 表示积木块 A 在其他积木块上面)类似本文使用的性质.在他们的方法下,这些领域控制规则被表示成易被 TLPlan 规划器使用的线性时态逻辑公式(LTL).

值得一提的是 Zhuo 等人[27]在规划库中提取有用知识方面的工作.他们的工作解决了如何从给定的规划库(PC)中尽可能地使用那些可以(部分)重用的知识.但他们的工作并没有涉及到如何获取相似规划,这会导致他们的方法会学习到过多无用知识,从而削弱他们的方法对规划求解效率提升的正面作用.

Gerevini 等人[28]讨论了如何从一个规划库(a library plan)中检索相似规划.他们把规划问题表示成一规划问题图(planning problem graph),并给出了他们的问题相似测度以及在规划问题图基础上计算问题相似匹配的核函数方法.他们的方法证实了在能够检索到相似问题规划解的前提下,规划重用比传统从零构造规划更有效率.但他们衡量问题相似的方法本质上是在考虑对象匹配的情况下的最大命题匹配,并没有脱离 Nebel 的理论框架.而我们的方法可以保证学习一定是在结构等价或者结构尽可能相似的问题之间.

我们所做的将学习到的先验知识进行合并编码成 SAT 句子的工作借鉴了 Mali 等人的工作[16].他们的工作主要是讨论如何在可满足框架下将若干个规划合并成一个全局规划以及重用规划.但他们的工作存在瑕疵.他们在规划合并方面的工作缺少了一部分约束,这会导致非预期的伪模型出现.与 Mali 等人的工作相比,我们的合并逻辑比他们的更简单,而且我们的合并约束考虑了不同规划之间存在相同动作的情况,这是 Mali 等人的规划合并工作所没有考虑的.

Van der Krogt[29]讨论了如何在可满足框架下将已有规划Δ_0进行修改,以得到当前问题Π的规划解Δ.但 van der Krogt 并没有讨论如何得到Δ_0,而如何得到相似问题的解是规划重用的难点,也是规划重用能否有效的关键.

7 结论及未来的进一步工作

本文给出了一种能够有效地组织和存储规划器求解经验知识,并在新问题求解时能够有针对性地回忆/检索相关经验知识,最后能够将这些学习到的知识有效地转换成可满足的句子.这些句子所代表的约束知识可有效地压缩搜索空间,从而加速 SAT 求解器的求解效率.理论和实验结果均证明了本方法的有效性.

从目前来看,在本文工作的基础上可进一步开展的工作主要包括以下 3 个方面:

(1) 先验知识的充分使用方面的工作.SOLP 系统只使用了先验知识中的动作之间的半序关系这一方面的知识.而先验知识中除动作之间的部分序关系这一知识外,还存在一个同样重要的知识:因果链约束知识并没有被 SOLP 系统加以充分使用.因此,如何进一步挖掘先验知识中所包含的因果链约束知识以进一步提高系统的效率,是本文下一步的重要工作.

(2) 先验知识的准确性方面的工作.SOLP 中通过子问题检索规划片段,而这些检索到的规划片段显然不一定与当前待求解问题的真正解相应的规划片段等价,而可能出现不相容的规划片段的情况,这就导致学习到无用知识的情况出现.因此,如何在不影响系统总体性能的基础上进一步提高先验知识的准确性,是值得研究的课题.

(3) 先验知识的使用方式方面的工作.SOLP 系统是在可满足框架下使用先验知识,另一个更为直观的使

用先验知识的方式是在部分规划框架下使用这些知识.在部分规划框架下,这些学习到的每条先验知识都可看作是一个部分规划.新问题的求解则是在这些先验知识为初始解的基础上进行搜索.

References:

[1] Rosa TDL, Jiménez S, Fuentetaja R, Borrajo D. Scaling up heuristic planning with relational decision trees. Journal of Artificial Intelligence Research, 2011,40(4):767−813. [doi: 10.1613/jair.3231]

[2] Fern A, Yoon S, Givan R. Approximate policy iteration with a policy language bias: Solving relational Markov decision processes. Journal of Artificial Intelligence Research, 2006,25(1):75−118. [doi: 10.1613/jair.1700]

[3] Fox M, Long D, Magazzeni D. Plan-Based policy-learning for autonomous feature tracking. In: Proc. of the 22nd Int'l Conf. on Automated Planning and Scheduling (ICAPS 2012). AAAI Press, 2012. 38−46.

[4] Yoon S, Fern A, Givan R. Learning control knowledge for forward search planning. The Journal of Machine Learning Research, 2008,9:683−718.

[5] Botea A, Enzenberger M, Müller M, Schaeffer J. Macro-FF: Improving AI planning with automatically learned macro-operators. Journal of Artificial Intelligence Research, 2005,24(10):581−621. [doi: 10.1613/jair.1696]

[6] Gerevini A, Saetti A, Vallati M. Exploiting macro-actions and predicting plan length in planning as satisfiability. In: Proc. of the AI*IA 2011. Berlin, Heidelberg: Springer-Verlag, 2011. 189−200.

[7] Newton M. Wizard: Learning macro-actions comprehensively for planning [Ph.D. Thesis]. Department of Computer and Information Science, University of Strathclyde, 2008.

[8] Bibai J, Savéant P, Schoenauer M, Vidal V. An evolutionary metaheuristic based on state decomposition for domain-independent satisficing planning. In: Proc. of the ICAPS. AAAI Press, 2010. 18−25.

[9] Vidal V, Geffner H. Branching and pruning: An optimal temporal POCL planner based on constraint programming. Artificial Intelligence, 2006,170(3):298−335. [doi: 10.1016/j.artint.2005.08.004]

[10] Dréo J, Savéant P, Schoenauer M, Vidal V. Divide-and-Evolve: The marriage of descartes and darwin. In: Proc. of the 7th Int'l Planning Competition—The Deterministic Part. Menlo Park: AAAI Press, 2011. 29−30.

[11] Wu JH, Givan R. Automatic induction of Bellman-error features for probabilistic planning. Journal of Artificial Intelligence Research, 2010,38(8):687−755.

[12] Gerevini A, Saetti A, Vallati M. An automatically configurable portfolio-based planner with macro-actions: PbP. In: Proc. of the ICAPS 2009. AAAI Press, 2009. 350−353.

[13] Nebel B, Koehler J. Plan reuse versus plan generation: A theoretical and empirical analysis. Artificial Intelligence, 1995,76(1-2): 427−454. [doi: 10.1016/0004-3702(94)00082-C]

[14] Fox M, Long D. The automatic inference of state invariants in TIM. Journal of Artificial Intelligence Research, 1998,9(1):367−421. [doi: 10.1613/jair.544]

[15] Fox M, Long D. The automatic inference of state invariants in TIM. CoRR abs/1105.5451, 2011.

[16] Mali AD. Plan merging & plan reuse as satisfiability. In: Proc. of the 5th European Conf. on Planning (ECP'99): Recent Advances in AI Planning. Springer-Verlag, 2000. 84−96. [doi: 10.1007/10720246_7]

[17] Kautz HA, Selman B. Planning as satisfiability. In: Proc. of the ECAI'92. John Wiley & Sons, Inc., 1992. 359−363.

[18] Ernst MD, Millstein TD, Weld DS. Automatic SAT-compilation of planning problems. In: Pollack ME, ed. Proc. of the 15th Int'l Joint Conf. on Artifical Intelligence (IJCAI'97), Vol.2. San Francisco: Morgan Kaufmann Publishers, 1997. 1169−1176.

[19] Kautz HA, McAllester DA, Selman B. Encoding plans in propositional logic. In: Proc. of the KR'96. Cambridge: Morgan Kaufmann Publishers, 1996. 374−384.

[20] Kautz HA, Selman B. Pushing the envelope: Planning, propositional logic and stochastic search. In: Proc. of the AAAI/IAAI, Vol.2. Menlo Park: AAAI Press, MIT Press, 1996. 1194−1201.

[21] Kautz HA, Selman B. Unifying SAT-based and graph-based planning. In: Proc. of the IJCAI. Morgan Kaufmann Publishers, 1999. 318−325.

[22] Blum A, Furst ML. Fast planning through planning graph analysis. In: Proc. of the IJCAI. AAAI Press, 1995. 1636−1642.

[23] Bacchus F, Kabanza F. Using temporal logics to express search control knowledge for planning. Artificial Intelligence, 2000, 116(1-2):123−191. [doi: 10.1016/S0004-3702(99)00071-5]

[24] Kvarnström J, Doherty P. TALplanner: A temporal logic based forward chaining planner. Annals of Mathematics and Artificial Intelligence, 2011,30(1-4):119−169. [doi:10.1023/A:1016619613658]

[25] Nau D, Au TC, Ilghami O, Kuter U, Murdock JW, Wu D, Yaman FS. SHOP2: An HTN planning system. Journal of Artificial Intelligence Research, 2003,20(1):379−404. [doi: 10.1613/jair.1141]

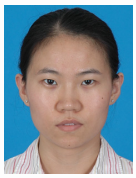
[26] De la Rosa T, McIlraith SA. Learning domain control knowledge for TLPlan and beyond. In: Proc. of the ICAPS 2011 Workshop on Planning and Learning (PAL). 2011. 36−43.

[27] Zhuo HK, Yang Q, Li L. Constraint-Based case-based planning using weighted MAX-SAT. In: Mcginty L, Wilson DC, eds. Proc. of the 8th Int'l Conf. on Case-Based Reasoning: Case-Based Reasoning Research and Development (ICCBR 2009). Berlin, Heidelberg: Springer-Verlag, 2009. 374−388. [doi: 10.1007/978-3-642-02998-1_27]

[28] Gerevini A, Saetti A, Serina I. Case-Based planning for problems with real-valued fluents: Kernel functions for effective plan retrieval. In: Proc. of the ECAI 2012. IOS Press, 2012. 348−353.

[29] van der Krogt RPJ. Modification strategies for SAT-based plan adaptation. Archives of Control Sciences, 2008,18(2):203−230.

陈蔼祥(1978－),男,广东兴宁人,博士,副教授,主要研究领域为智能规划,自动推理,统计机器学习.
E-mail: cax413@163.com



边芮(1982－),女,博士,讲师,主要研究领域为智能规划.
E-mail: bianrui61@163.com

姜云飞(1945－),男,教授,博士生导师,主要研究领域为智能规划,自动推理.
E-mail: issjyf@mail.sysu.edu.cn

陈清亮(1980－),男,博士,副教授,CCF 会员,主要研究领域为人工智能,自动推理.
E-mail: tpchen@jnu.edu.cn

柴啸龙(1980－),男,博士,讲师,主要研究领域为智能规划.
E-mail: chaixiaolongok@163.com