基于因果图启发式的并行概率规划求解*

饶东宁[1], 朱永亮[1], 蒋志华[2†]
(1. 广东工业大学 计算机学院, 广州 510006; 2. 暨南大学 信息科学技术学院 计算机科学系, 广州 510632)

摘 要: 并行概率规划(PPP)是近年来智能规划领域中的研究热点。在该类问题中,动作具有并发性和不确定性,非常贴近现实问题。然而现有的两种针对 PPP 的主要求解方法都有明显的缺点。因此,尝试使用高效的启发式搜索方法来求解这类问题。考虑到 PPP 问题采用 RDDL 语言来描述,其中的条件概率函数(CPF)非常适合用于构建因果图(CG),所以引入因果图启发(CGH)来进行求解。提出的启发式算法称为 CGH_{RDDL},整体求解方法是使用 rddlsim 模拟状态演化以及用 CGH_{RDDL} 引导搜索。实验结果表明,在不允许手工干预和参数调整的前提下,该方法的求解效果要好于代表性规划器 PROST 和 Glutton;并且与其他启发式相比,CGH_{RDDL} 的求解质量高于随机搜索,求解速度快于爬山法,这表明在经典规划领域中高效的启发式搜索策略可扩展去求解这一类非经典规划问题。

关键词: 并行概率规划; 因果图; 领域转换图; 因果图启发
中图分类号: TP3 **文献标志码**: A **文章编号**: 1001-3695(2018)05-1372-08
doi:10.3969/j.issn.1001-3695.2018.05.020

Solving parallel and probabilistic planning problems based on causal graph heuristic

Rao Dongning[1], Zhu Yongliang[1], Jiang Zhihua[2†]
(1. *School of Computers, Guangdong University of Technology, Guangzhou* 510006, *China*; 2. *Dept. of Computer Science, School of Information Science & Technology, Jinan University, Guangzhou* 510632, *China*)

Abstract: Parallel and probabilistic planning (PPP) is a hot spot in AI planning in recent years. In the PPP, actions could be concurrent and non-deterministic. These properties were very close to those of practical problems in the real world. However, both of the two primary methods of solving the PPP had obvious shortcomings, respectively. Therefore, this paper tried to apply some efficient heuristic searching methods into solving the PPP. The newest planning description language for the PPP was the RDDL where the conditional probabilistic functions (CPF) were very helpful to construct causal graphs (CG). Therefore, it introduced causal graph heuristic (CGH) into solving the PPP. Its core heuristic algorithm was called CGH_{RDDL}. In the solving process, it used the simulator, rddlsim, to simulate the state evolution and used the CGH_{RDDL} to guide the searching. Experiment results in some PPP benchmark domains show that the proposed method performs better than the two representative planners, PROST and Glutton, without allowing any manual intervention or parameter resetting. Furthermore, when compared with other heuristic methods, the CGH_{RDDL} is better than the random searching method and faster than the pure hill-climbing method. This shows that it can extend some efficient heuristic searching methods in the classical planning to solve this kind of non-classical planning problems.

Key words: parallel and probabilistic planning(PPP); causal graph(CG); domain transition graph(DTG); causal graph heuristic(CGH)

0 引言

并行概率规划(parallel probabilistic planning, PPP)[1~3]是目前智能规划领域研究的复杂规划问题之一。它具有并行规划和不确定规划的组合特征,允许概率动作并发地执行,动作变量和状态变量满足全局性约束,非常贴近现实问题。规划领域对 PPP 非常重视,在各个层面都投入了关注,包括提出新的语言[4]、开发新的规划器[5,6]以及建立新的比赛。针对 PPP,现有两种主要求解方法。一种方法基于 UCT 算法[7],以 PROST 系列[5]的规划器为代表。UCT 算法是个蒙特卡罗树抽样过程,在极限情况下是最优的,但需要较长时间才能达到收敛。因此,PROST 的求解质量虽好,但是速度太慢,对很多基准问题实例在限定的 30 min 内无法求解。另一种方法基于 LRTDP 算法[8],以 Glutton 系列[6]的规划器为代表。Glutton 采用迭代加深的方式来运行 RTDP。对 H 步有限域问题,它先解决一个 1 步版本,然后产生一个 2 步版本,如此循环。然而尽管 Glutton 的求解速度快,但是其求解质量远不如 PROST。因此,两种主要的 PPP 求解方法各有不足,需要新的方法来同时兼顾求解质量和求解效率。

启发式搜索是一种高效的人工智能求解方法。在规划领域里,基于启发式的规划方法获得了里程碑式的成功[9,10]。其基本思想很简单,设计启发式函数对备选的后继状态或者部分规划进行评估,选取最有希望的候选者进行扩展。但构建合适

收稿日期: 2016-12-27; **修回日期**: 2017-02-20 **基金项目**: 广东省自然科学基金资助项目(2016A030313084, 2016A030313700, 2014A030313374);中央高校基本科研业务费专项资金资助项目(21615438);广东省科技计划资助项目(2015B010128007)

作者简介:饶东宁(1977-),男,广东兴宁人,副教授,博士,主要研究方向为智能规划;朱永亮(1989-),男,硕士研究生,主要研究方向为智能规划;蒋志华(1978-),女(通信作者),副教授,博士,主要研究方向为智能规划(tjiangzhh@jnu.edu.cn).

的启发式函数并不是容易的事情,因为既要保证启发估值的准确性,又要不存在太大的计算代价。启发式规划特别适用于求解大规模、要求效率、不要求最优解的规划问题。在目前的规划研究领域中,有三种基于启发式的规划系统最具有代表性,它们是基于放松规划图启发的 FF[11]、基于可加式启发的 HSP[12]、基于因果图启发的 Fast Downward[13,14]。事实上,前两种规划器都使用放松式规划图启发,而后一种规划器采用因果图和领域转换图来计算启发值。

为了更快更好地求解 PPP 问题,本文考虑引入因果图启发式(causal graph heuristic,CGH)搜索算法。这是因为在 PPP 领域描述中的条件概率函数(conditional probabilistic function,CPF)[4,15],即状态变量取值的转换规则,提供了构建因果图(causal graphs,CG)和领域转换图(domain transition graphs,DTG)的良好平台。CG 描述了一个规划领域中不同状态变量之间的依赖关系,而 DTG 则刻画了同一状态变量的不同取值之间的转换条件。两者的关系是 CG 提供了高层任务的分解,而 DTG 则进一步指出如何实现这些高层任务之间的抽象步骤。CGH 算法需要同时使用到这两个图。它们的构建来自于因果链(causal link,CL),只要确定了因果链,就可以很方便地进行构建。例如,Fast Downward 使用动作模型中前提—效果以及共生效果来产生因果链[13]。为了更方便地表示并发性和不确定性,PPP 问题采用新的规划语言 RDDL(relational dynamic influence diagram language)来描述领域和问题实例,而非传统的动作模型。RDDL 采用 CPF 去描述流文字之间的互相影响,能表征更宽泛的依赖关系,而不限制于单纯的前提与效果之间。因此,借助于 RDDL 描述,构建 CG 和 DTG 以及引入 CGH 搜索算法是可行的。

但是 CGH 算法不能直接应用于 PPP 问题,需要进行改进。首先,最主要的原因在于,PPP 问题一般都没有固定的目标状态,这使得在 DTG 上状态变量的目标代价估算无从做起。PPP 问题的求解目标是在有限的轮数内或资源的限制下使得 Reward 函数达到最优。因此,状态的启发值应包含通过有限步骤所产生的状态轨迹的回报值,而非相对于固定目标状态的距离值。其次,RDDL 中的 CPF 非常灵活。由于动作和状态均表示成流文字,所以同一个 CPF 的条件部分可出现多个动作流文字,这体现了动作的并发性。同时,CPF 还可以描述状态流文字成立的概率。这些概率值往往作为 Reward 函数的一部分,参与启发值的计算。简言之,CPF 能实现公理、动作模型以及概率效果等多种领域特征。再次,RDDL 还包括了动作或状态的全局性约束,其中最常见的是唯一性需求。例如,对象在同一时刻只能应用一个动作或者只能位于一个地方。这要求在产生后继状态时,需检测全局性约束是否满足。最后,在 PPP 中,由于动作的并发性,对象之间的交互和协作更加频繁,而在 h^1[12] 或者 SAS + −1[13] 等高效的启发式中往往假设子问题间是相互独立的。这种假设可以快速地算出启发值,但是由于忽略了对象间的交互而损失了准确性。因此,如何在 PPP 的启发式估值中适当地引入子问题间的交互是需要考虑的问题。

针对以上问题,本文提出了基于 CGH 来求解 PPP 问题的启发式规划方法,称为 CGH_{RDDL}。具体步骤如下:

a)根据 RDDL 描述的 CPF 构建 CG 和 DTG。CG 反映了不同状态变量之间的依赖关系,针对领域,只有一个;而 DTG 反映同一状态变量不同取值之间的转换关系,需对每个状态变量进行构建。可从领域描述中构建 DTG 的统一结构,但在具体估值时需根据问题实例对 DTG 进行实例化。

b)根据 CG 和 DTG,计算单个状态变量任意一对取值间的转换代价。这是状态启发值的一部分,但可以在预处理阶段完成。这些转换代价只需计算一次,而后反复使用。

c)使用状态检测和演化的官方工具 rddlsim 来模拟求解,直至规定的轮数。在这个过程中,每一步使用 CGH_{RDDL} 来估算后继状态的启发值,选取最佳状态进行前向演化。其中,状态的启发值以 Reward 函数为基础,还包括转换代价和折扣因子等。

d)累加在限定轮数内 rddlsim 状态演化的回报值,即为最终的求解质量。在 PPP 部分基准领域的实验结果表明,在无人工干预和参数调整的前提下,CGH_{RDDL} 的求解性能优于 PROST 和 Glutton。而且相对于其他启发式而言,CGH_{RDDL} 的求解速度高于普通的爬山法,求解质量高于随机搜索算法。

1　研究背景

1.1　并行概率规划

经典规划的假设不满足解决实际问题的需求,因此非经典规划领域成为目前智能规划研究的一个主要研究方向[16,17]。PPP 是近年来研究较多的非经典规划问题之一。它具有并行规划和不确定规划的组合特征,允许动作并发执行,且满足全局性约束,放松了经典规划的五大假设[1]。针对 PPP,规划社区提出了新的描述语言 RDDL[4]、开发新的规划器(如 PROST 和 Glutton)以及建立新的规划比赛 IPPC(international planning probabilistic competition)。PPP 比较贴近现实问题,在 IPPC 上的大多数基准领域都是现实问题的简化模型。例如,elevator 领域描述多个电梯调度问题,乘客到达每个楼层具有概率分布,电梯调度正确则给予一定的奖赏,反之则获得惩罚。再如,traffic 领域描述交通灯控制问题,通过动作可以控制十字路口的交通灯信号,从而尽量避免车流拥塞。

目前来说,关于 PPP 的规划竞赛有 IPPC-2011(http://users. cecs. anu. edu. au/ ~ ssanner/IPPC_2011/) 和 IPPC-2014(https://cs. uwaterloo. ca/ ~ mgrzes/IPPC_2014/index. html)。IPPC-2014 的竞赛领域和实例设置与 IPPC-2011 基本相同,区别主要是在比赛规则上。在 IPPC-2011 上,允许手动调参或者与规划器交互,如终止运行、调试错误、改变参数、重新启动等。而在 IPPC-2014 上,比赛开始后则不允许手动调整参数,且每个问题实例的运行有时间限制。IPPC 采用 RDDL 来描述 PPP 领域,这是因为动作的并发性以及全局性约束使得基于效果的动作模型不再适用,即 PDDL 系列[18],甚至包括 PPDDL[19] 都不适用于描述 PPP。从语义上讲,RDDL 是一个关系动态贝叶斯网络(relational dynamic Bayesian network,RDBN)[20]。RDDL 具有许多新的特性。例如,一切都是变元,包括状态、动作和观测;变元可以是一个流文字,也可以是一个非流文字;报偿函数是一个概率分布或者表达式,允许包括各种聚集函数;CPF 代替动作模型用来描述状态变量的值如何改变等。这些特性使得用 RDDL 来描述 PPP 问题得心应手。当然,RDDL 并非全新的语言,其主要元素与 PDDL 系列仍有着密切的联系。例如,CPF 非常类似于 PDDL2.2[21] 中的公理或者派生谓词规则。再如,其强大而灵活的约束表达式可追溯到 PDDL 3.0[18] 中的简单约束。

求解 PPP 问题的困难在于其状态空间异常庞大。这是因为在传统的规划方法中,当不确定性和并行性并存时,任何一组不确定效果的组合都将产生一个可能的后继状态[1]。整个后继状态集合的势等于不确定效果数的笛卡尔积。因此在搜索或演化的每一步,PPP 都会产生许多后继状态,通过穷举状态或者可达路径分析来进行估值是极其困难的。再加上状态空间呈现一定的概率分布,概率的演算大大增加了 PPP 求解的复杂性。在已有的两届 IPPCs 中,PROST[5] 和 Glutton[6] 相对于其他规划器表现得更好,因此它们代表了目前主要的 PPP 求解方法。PROST 改进了 UCT,它用一个简单的确定化方法来计算两步的前向启发式,并保存所有计算过的状态—动作对。参加 IPPC-2014 的新版本 PROST 2014 使用了更多的启发式,如修剪不合理动作、报偿锁检测和基于迭代加深的方法等。然而 UCT 算法的固有缺陷还是使得它的求解速度过慢。Glutton 基于 MDP 的求解过程,使用 LR2TDP 算法。其中用来处理复杂问题实例的最重要的优化算法是二次抽样转换函数,它可分离出高速缓存转移函数的样本。由于 RTDP 在贝尔曼备份过程中要枚举动作,所以往往失败。Glutton 用状态动作采样来解决这个问题,并缓存采样的动作成果。为了便于实时规划,Glutton 采用迭代加深的方式来运行 RTDP。这些技术使得 Glutton 的求解速度较快,但是在解质量方面并没有什么优化措施。

1.2 因果图启发式规划

在规划研究领域中,有三种基于启发式的规划系统最具有代表性,它们是 FF、HSP、Fast Downward。FF 忽略动作的删除效果,得到一个放松规划图,通过该图能快速得到后继状态的启发值。由于增加了 helpful actions 以及 goal agenda 等技术,FF 的求解速度非常快。HSP 采用 h^+ 启发式,由 h^1,h^2,…等一系列的启发式函数构成(其中最常用的是 h^1 和 h^2)。状态的启发值等于求解子任务的最小代价之和,每个子任务可由多个目标组合。其可加式的本质是忽略子任务之间的相互影响,假设问题独立来快速获取启发值。与 FF 和 HSP 不同的是,Fast Downward 先采用因果图来进行层次任务分解,然后在领域转换图上估算单个状态变量距离目标值的代价。因此,这两种图的构建对于 Fast Downward 的启发值计算非常重要。但是目前还无法自动地从领域描述中构建图,只能手工进行。同时,Fast Downward 借鉴了前辈们的成果,像 FF 一样选取偏好动作,像 HSP 一样采用可加式求和。这些技术使得 Fast Downward 以及后继版本在国际规划大赛(international planning competition,IPC)中取得了很好的成绩。

然而基于放松式规划图的启发在某些情况下会产生误判,将不可解状态(即 dead ends)计算成可解状态。例如,图 1 展示了一个 Depot 领域的问题实例:要将货物从 E 运到 B(如虚线所示),两辆货车分别停在 C 和 F。由于 D 只有入边没有出边,所以该问题是无解的。但是在计算其放松式问题的启发值时,会得到 8,而不是 ∞。导致误判的原因在于对问题的放松,即忽略了动作执行的删除效果。这导致了当货车在移动时会形成无法消除的残影。最终经过移动,同一辆货车变成了在移动路径上不断复制,就像具有“瞬间移动”的功能一样,使得这个 dead end 变成了可解状态。

采用因果图启发可以消除这样的误判。因为它不忽略动作的删除效果,而是把规划问题拆分成一系列的单目标子任务来进行求解,累加求解代价从而得到启发值。具体方法如下:a)把规划问题的二值描述形式转换成多值规划描述(即 SAS+);b)根据问题描述,构建因果图和领域转移图,并根据这两个图计算任意状态变量两个取值之间的转换代价;c)根据规划问题的目标,将原问题拆分为一系列 SAS+ −1 子任务(1 表示出现在目标中的单个状态变量),通过从当前状态求解这些子任务来估算状态的启发值。在计算启发值时,为了避免循环转换,可能需要修剪因果图使其成为无环图,这会破坏原本的依赖关系。所以各个子任务的求解结果的合并不能得到原有问题的可行策略,但是它们求解代价的简单加和却可以得到高质量的启发值。

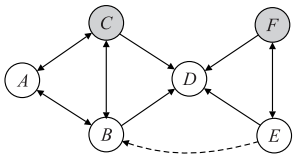

图1 放松式规划图启发误判的例子

2 问题陈述

2.1 RDDL

RDDL 描述由六个部分组成(图 2)。其中,requirements 指定领域的特征和类型;types 定义对象的类型;pvariables 定义领域中所有变量,包括动作变量、状态变量和观测变量;cpfs 是条件概率函数,定义状态变量的取值如何变化;state action constraints 是全局性约束,规定了状态变量或者动作变量的取值必须满足的约束;reward 是报酬函数,定义了状态的立即回报值。cpfs 和 reward 是两个最主要的成分。cpfs 代替了传统的动作模型,而 reward 描述了目标的最优化机制。下面详细介绍 RDDL 的各个组成要素。

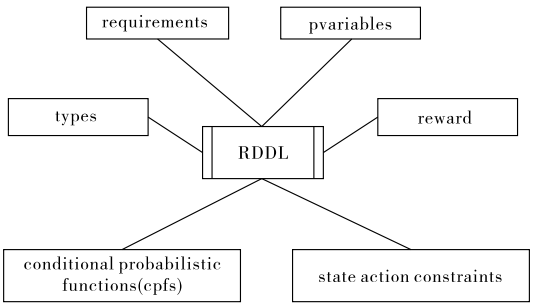

图2 RDDL的组成要素

定义 1 RDDL 领域描述。一个 RDDL 领域文件是一个六元组,即 $\Sigma_{\text{RDDL}}=\langle RE,T,PV,\text{CPFs},R,\text{SAC}\rangle$。其中:

a)RE 表示需求 requirements,包括 continuous、multivalued、partially-observed 等在内的九种具体需求。

b)T 表示类型定义 type definitions,所有类型都属于 object 类型。

c)PV 表示状态变量 pvariables,分为流文字(fluent)和非流文字(non-fluent)两种。流文字的本质是一种关系,其真值随着状态变化;而非流文字的真值在任意状态中保持不变,实质上代表了常量。流文字进一步分为状态流文字(state-fluent)、动作流文字(action-fluent)、中间流文字(interm-fluent)和可观测流文字(observ-fluent)四种。每个状态变量的声明分为是否流文字、类型以及缺省值三个部分。

d)CPFs 表示条件概率函数 conditional probabilistic func-

tions,描述状态变量的取值如何变化。每个 CPF 表达式可由常量、逻辑表达式、算术表达式、关系表达式、条件表达式和基本的概率分布函数等来构成。其中,逻辑表达式允许量词和各种逻辑算子;算术表达式允许基本的数学运算以及累加(sum)、累乘(product);条件表达式使用 if-then-else 和 switch 结构;基本概率分布包括伯努利分布、正态分布、泊松分布等。

e)R 表示报酬 reward,由包含状态变量的数学函数组成,表示当前状态的奖励值或惩罚值的累积。

f)SAC 表示 state action constraints,由逻辑表达式构成。其规定了状态流文字或者动作流文字的取值必须满足的约束,是全局性约束,即在任意时刻或任意状态中必须满足的约束。

下面以 IPPC 2011 的 Elevator 领域为例,看其 RDDL 描述的 cpfs 和 reward 是怎样定义的。Elevator 领域描述了乘客搭乘电梯到达目的楼层的问题。有多个电梯可调度,乘客到达每层楼有一定的概率分布,乘客的目的地只能是顶楼或者底楼。该领域有两种对象类型,一种是电梯 elevator,另一种是电梯能到达的楼层 floor。而具体电梯和楼层的数目由问题实例给出。

例 1　Elevator 领域的部分 cpfs 描述如下。该领域共有七个 cpfs,每个 cpf 均采用条件语句结构,其中关于两个方向(即 up 和 down)的 cpfs 呈对称结构。在以下示例中,? e 和? f 表示对象变元,KronDelta 表示断言,Bernoulli 表示伯努利分布。由于状态变量的真值可能会受到原有真值的影响,所以对同一状态变量,用 p 和 p'来区分当前状态的状态变量和下一状态的该变量(如 person-waiting-up 和 person-waiting-up′)。可以看到,有些 cpf 相对简单(如定义 person-waiting-up′(? f)的),而有些 cpf 却比较复杂(如定义 elevator-at-floor′(? e, ? f)的)。关于 person-waiting-up′(? f)的 cpf 表明,假设原来有乘客在该楼层等候,如果没有电梯把他(们)接走,则在下一状态他(们)依然在该楼层等候;或者如果有电梯把原有乘客接走了,但新乘客可能会到达,则在下一状态乘客等候的概率取决于到达的概率。当然,乘客上电梯的充要条件是电梯运行方向与其目的地方向一致且电梯门打开。而关于 elevator-at-floor′(? e, ? f)的 cpf 看似很复杂,但是从语义上讲很简单,即电梯上移一层或者下移一层,可以使得电梯停留在某层。

```
cpfs {
  person-waiting-up'(? f) =
  if (person-waiting-up(? f) ^
   ~exists_{? e: elevator} [elevator-at-floor(? e,? f) ^ elevator-
dir-up(? e) ^ ~elevator-closed(? e)])
  then KronDelta(true)
  else Bernoulli(ARRIVE-PARAM(? f));
  elevator-at-floor'(? e, ? f) =
  if (~elevator-closed(? e) | ~move-current-dir(? e))
  then KronDelta( elevator-at-floor(? e, ? f) )
  else if (move-current-dir(? e) ^ elevator-dir-up(? e) ^ exists_
{? cur:floor} [elevator-at-floor(? e, ? cur) ^ ADJACENT-UP(? cur,
? f)])
  then KronDelta(true)
  else if (move-current-dir(? e) ^ ~elevator-dir-up(? e) ^ exists_
{? cur : floor} [elevator-at-floor(? e, ? cur) ^ ADJACENT-UP(? f,
? cur)])
  then KronDelta(true)
  else if (move-current-dir(? e) ^ elevator-dir-up(? e) ^ ~exists_
{? next : floor} [elevator-at-floor(? e, ? f) ^ ADJACENT-UP(? f,
? next)])
  then KronDelta( elevator-at-floor(? e, ? f) )
  else if (move-current-dir(? e) ^ ~elevator-dir-up(? e)^~exists_
{? next: floor} [elevator-at-floor(? e,? f) ^ ADJACENT-UP(? next,
? f)])
  then KronDelta( elevator-at-floor(? e, ? f) )
  else KronDelta( false );
};
```

例 2　Elevator 领域的 reward 函数定义如下。由于没有固定的目标状态,该领域问题的求解目标是在规定的轮数内或者在限定的时间内使得 reward 函数最大化。Reward 函数分为两部分,即电梯运行方向正确与否的惩罚值和在楼层等待电梯的乘客的惩罚值。如果电梯运行方向不正确(即与电梯中乘客的目的地方向不一致),则给予惩罚值 ELEVATOR-PENALTY-WRONG-DIR;反之给予惩罚值 ELEVATOR-PENALTY- RIGHT-DIR。这是因为目标是尽可能多地将乘客送到其目的楼层,所以即使电梯运行方向正确,但电梯里有乘客也要给予一定的惩罚值。而在某个楼层如果有乘客等电梯,也要给予一定的惩罚值(惩罚值为 1),因为这时乘客并没有到达目的地。总的来说,只有乘客到达其目的楼层,才不会获得惩罚值。

```
reward =
  [sum_{? e: elevator} [
  -ELEVATOR-PENALTY-RIGHT-DIR * (person-in-elevator-going-
up(? e) ^ elevator-dir-up(? e))]] +
  [sum_{? e: elevator} [
  -ELEVATOR-PENALTY-RIGHT-DIR * (person-in-elevator-going-
down(? e) ^ ~elevator-dir-up(? e))]] +
  [sum_{? e: elevator} [
  -ELEVATOR-PENALTY-WRONG-DIR * (person-in-elevator-go-
ing-up(? e) ^ ~elevator-dir-up(? e))]] +
  [sum_{? e: elevator} [
  -ELEVATOR-PENALTY-WRONG-DIR * (person-in-elevator-
going-down(? e) ^ elevator-dir-up(? e))]] +
  [sum_{? f: floor} [
  -person-waiting-up(? f) - person-waiting-down(? f)
  ]];
```

2.2　因果图启发式

在智能规划中,因果图这个术语最初由 Williams 和 Nayak 提出,是表示所有状态变量层次(依赖)关系的一个有向图。它简洁地描述了各不同变量之间的依赖关系,使得本文可以更好地利用问题本身的结构信息来优化问题的求解过程。因果图是因果图启发式搜索的一个核心概念。本文首先给出因果图的相关定义。定义 2 ~ 6 均来自文献[13]。

定义 2　SAS + 规划任务是一个四元组 $\Pi = \langle V, O, s_0, s_g \rangle$:

a)$V = \{v_1, \cdots, v_n\}$ 是状态变量集合,每个状态变量 v 具有有限论域 D_v;

b)O 是操作集合,每个操作 $o = \langle \text{pre}, \text{eff} \rangle$ 具有前提 pre 和效果 eff;

c)s_0 是初始状态,s_g 是目标条件。

如果 s 定义在 V 的子集上且 $s(v) \in D_v$,则 s 称为部分(变量)赋值函数。如果对于任意 $v \in V$,$s(v)$ 有定义,则 s 称为状态。在定义 2 中,pre、eff 和 s_g 都是部分赋值函数。

定义 3　设 $\Pi = \langle V, O, s_0, s_g \rangle$ 是 SAS + 规划任务,则其因果图(CG)是一个有向图(V, A)。其中,V 是 Π 中所有状态变量对应的顶点集合。如果存在边$\langle u, v \rangle \in A$,当且仅当 $u \neq v$,且存在操作 $o = \langle \text{pre}, \text{eff} \rangle \in O$,使得下列条件之一成立:

a)u 是 o 的前提且 v 是 o 的效果,即 pre(u)和 eff(v)有定义;

b)u 和 v 是 o 的共生效果,即 eff(u)和 eff(v)有定义。

定义 4　对于状态变量 $v \in V$,其领域转换图(DTG)G_v 是定义在 D_v 上的标志有向图。对于任意 d, $d' \in D_v$,如果存在有向边$\langle d, d' \rangle$,当且仅当存在着 $o = \langle \text{pre}, \text{eff} \rangle \in O$,使得 eff($v$) = d',且 pre(v) = d 或者 pre(v)无定义。边$\langle d, d' \rangle$的标志为 L = pre|($V \backslash \{v\}$),即 pre 中除 v 之外的其他变量赋值。如果存在

着这样的一条边,则称为存在着一个在条件 L 下关于变量 v 的从 d 到 d' 的转换。

在本文方法中需根据 cpfs 画出 CG 和 DTG。简单来说,在构建 CG 时,对于每个 cpf,从 cpf 等号后边的每个 fluent 在 CG 上创建一条有向边指向等号左边的 fluent,其中 state_fluent 用椭圆框表示,action_fluent 用方框表示。用不同的图形表示这些流文字,是为了方便在 CG 上识别 cpf。在构建 DTG 时,为每个 cpf 等号左边的状态变量 v 创建一个 DTG_v。其中,v 的值域 D_v 为{true,false},这是因为 v 是 fluent 的缘故。对于出现在 cpf 右边的每个 fluent,如果其为正文字,则在转换边上的标记为 true;反之,如果是负文字,则标记为 false。对于一个规划领域,CG 只有一个,而 DTG 则可有多个,每个对应一个状态变量。

例如,Elevator 领域的 CG 如图 3 所示。从例 1 的 cpfs 得到的两个 DTG 如图 4 和 5 所示。

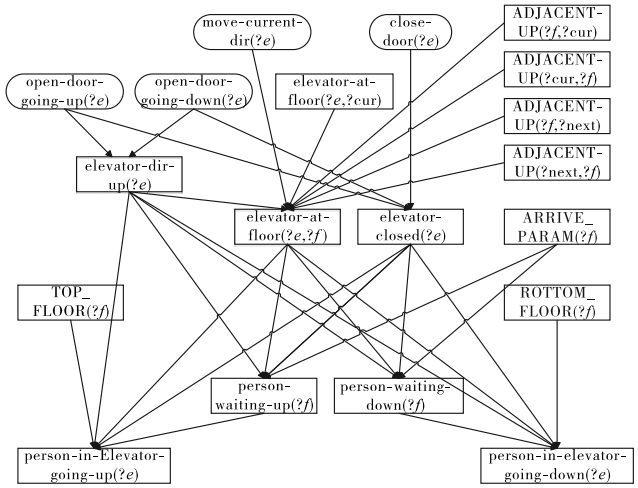

图3 Elevator领域的CG

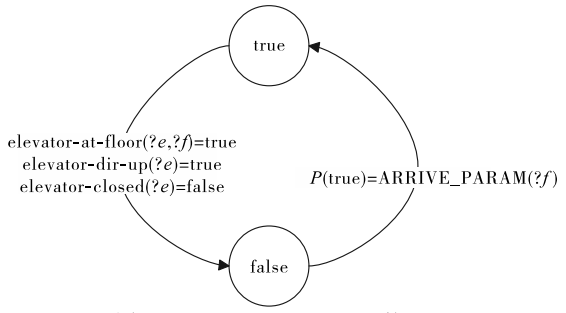

图4 Person-waiting-up'(?f)的DTG

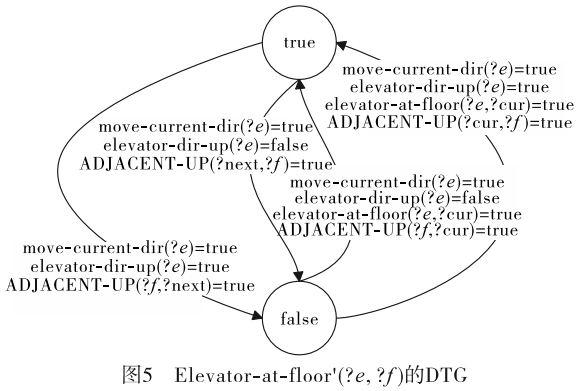

图5 Elevator-at-floor'(?e, ?f)的DTG

定义 5 因果图启发式函数 $h^{\mathrm{CG}}(s)$ 用于估计从给定状态 s 到目标状态 s^* 需用的动作数,即对于每个出现在目标中的状态变量 v,将其值从 $s(v)$ 变化到 $s^*(v)$ 的代价之和为

$$h^{\mathrm{CG}}(s) \triangleq \sum_{v \in \mathrm{dom}(s^*)} \mathrm{cost}_v(s(v), s^*(v)) \tag{1}$$

要计算 $\mathrm{cost}_v(s(v), s^*(v))$,必须知道其父变量 v_i 的代价 $\mathrm{cost}_{v_i}(e_i, e'_i)$, $\forall e_i, e'_i \in D_{v_i}$。对于任意边 $(d_1, d_2) \in \mathrm{DTG}(v)$,若边上标记的条件为 z,则

$$\mathrm{cost}_v(d_1, d_2) = 1 + \sum_{(v_i = e_i) \in z} \mathrm{cost}_{v_i}(e'_i, e_i) \tag{2}$$

定义 6 可加式启发函数 $h^{\mathrm{add}}(s)$ 用于计算 s 的启发值。

$$h^{\mathrm{add}}(s) \triangleq \sum_{x \in s^*} h^{\mathrm{add}}(x|s) \tag{3}$$

其中:$h^{\mathrm{add}}(x|s)$ 表示从 s 实现 x 的代价,即

$$h^{\mathrm{add}}(x|s) \triangleq \begin{cases} 0 & \text{if } x \in s \\ \min\limits_{o;z \to x} \left(1 + \sum\limits_{x_i \in z} h^{\mathrm{add}}(x_i|s)\right) & \text{if } x \notin s \end{cases} \tag{4}$$

其中:o 表示能实现 x 的操作;z 是其上标记的条件。

3 方法

3.1 求解算法整体框架

为了求解 RDDL 描述的 PPP 问题,RDDL 的提出者 Sanner 给出了一个模拟器 rddlsim(http://code.google.com/p/rddlsim/)。该模拟器可以解析 RDDL 的输入文件,进行状态演化,并依据概率分布产生随机的后继状态。不同于 PROST 和 Glutton,本文的 PPP 求解方法依赖于 $\mathrm{CGH}_{\mathrm{RDDL}}$ 算法与 rddlsim 的协作。下面先给出整体的规划求解流程(算法 1),然后再给出其中用于计算转换代价的算法(算法 2)和计算启发值的核心算法 $\mathrm{CGH}_{\mathrm{RDDL}}$(算法 3)。

算法 1 整体规划求解算法

输入:领域文件 domain.rddl、实例文件 instance.rddl。

输出:规划解 cpf_sol、报酬值 reward。

a)预处理

(a)构建 CG。对于任意 cpf,从 cpf 等号后边的每个 fluent 在 CG 上创建一条有向边指向等号左边的 fluent。

(b)构建 DTG。对于任意 cpf,为 cpf 等号左边的状态变量 v 创建一个 DTG_v。v 的值域 D_v 为{true,false}。对于出现在 cpf 右边的每个 fluent,如果其为正文字,则在转换边上的标记为 true;反之,如果是负文字,则标记为 false。

(c)计算转换代价。根据算法 2,对任意出现在 R 中的流文字 v,依据其 DTG_v,计算其任意一对值间转换的代价值,即 $\mathrm{cost}_v(\mathrm{false},\mathrm{true})$ 和 $\mathrm{cost}_v(\mathrm{true},\mathrm{false})$。

b)初始化。$h=1$, $s \leftarrow I$, cpf_sol $=\varnothing$, reward $=0$。

c)状态演化至限定轮数 H。

(a)当 h 未超过 H 时;

(b)根据 CG 生成后继状态集合 $S'=\{s'\}$ 应用一条至少包含一个 action_fluent 的 cpf 到 s 产生 s'};

(c)根据算法 3(即 $\mathrm{CGH}_{\mathrm{RDDL}}$ 算法),计算每个后继状态 $s' \in S'$ 的启发值;

(d)选择启发值最大的后继状态 s_{best},并记录下其对应的 $\mathrm{cpf}_{\mathrm{best}}$;

(e)调用 rddlsim 的 evolve(s, $\mathrm{cpf}_{\mathrm{best}}$)方法对 s 进行演化,得到下一步随机状态 s_{next};

(f)cpf_sol({$\mathrm{cpf}_{\mathrm{best}}$});

(g)reward $+= R(s_{\mathrm{next}}) * h'$;

(h)h 递增;

d)输出 cpf_sol 和 reward。

算法 1 的核心思想是:在用 rddlsim 对状态进行演化之前,先预测哪个可能的 cpf 应用会带来最大的报酬值。rddlsim 的作用在于,对于当前状态和指定的 cpf,它会生成一个后继状态的概率转移矩阵,然后依据概率大小从中随机产生一个后继状态,从而继续演化。因此,设计良好的启发式函数,会有希望在 rddlsim 的状态演化中产生最大的报酬值。领域文件 Domain.rddl 提供了一个领域定义 $\Sigma_{\mathrm{RDDL}} = \langle RE, T, PV, \mathrm{CPFs}, \mathrm{SAC}, R \rangle$,实例文件 Instance.rddl 提供了一个问题实例定义 $\Pi = \langle \Sigma_{\mathrm{RDDL}},$

$O,H,I\rangle$。其中,CPFs 是构建 CG 和 DTG 的主要依据;R 既用于计算状态轨迹的报酬值,又用于计算后继状态的启发值;由于没有固定目标状态,H 指定了状态演化的最大轮数;I 是初始状态;O 是对象集合,在需要时(如将多元谓词转换为多值状态变量时)可实例化 DTGv。

3.2　启发值算法

算法 2 用于计算状态变量 v 的取值从 d_1 到 d_2 的转换代价,即 $\mathrm{cost}_v(d_1,d_2)$。它的核心思想是:如果存在着边直接导致 d_1 变化到 d_2,则转换代价等于边上的转换条件的代价之和;如果不存在着直接边但存在着转换路径,则转换代价等于所有转换路径的最小代价。其中,转换路径的代价等于路径上各边的转换代价之和。算法 2 在 Fast Downward 计算转换代价的基础上,根据 RDDL 领域描述的特点作了改进。在 RDDL 中,由于同一 cpf 中允许动作并发执行,所以所有 action_fluent 都参与代价的计算。

算法 2　计算 $\mathrm{cost}_v(d_1,d_2)$

输入:状态变量 $v,d_1,d_2\in D_v$。

输出:$\mathrm{cost}_v(d_1,d_2)$。

a)初始化。$\mathrm{cost}_v(d_1,d_2)\leftarrow\infty$。

b)在 DTG_v 上,如果存在边 $d_1\rightarrow d_2$(label L),则

$\mathrm{cost}_v(d_1,d_2)=\sum_{k\in L,k\neq a}\mathrm{cost}_k(s(k),L(k))+|\{k\}|_{k\in L,k=a}$

其中,k 若是 action_fluent,则加 1;k 若是 state_fluent,则加上 k 在当前状态 s 的取值 $s(k)$ 到 L 中取值 $L(k)$ 的转换代价。

c)如不存在边 $d_1\rightarrow d_2$(label L),但存在路径 $p:d_1\rightarrow\cdots\rightarrow d_2$,则

$\mathrm{cost}_v(d_1,d_2)=\min\{\mathrm{cost}_P(d_1,d_2)\mid$在 DTG_v 上存在路径 P 从 d_1 到 $d_2\}$。

其中,$\mathrm{cost}(d_1,d_2)=\sum_{i=1}^{|P|}\mathrm{cost}_v(d_{k_i},d_{k_{i+1}}),d_{k1}=d_1,d_{k|P|+1}=d_2$。

算法 3 用来计算状态的启发值。给定当前状态 s 和可应用于 s 的 cpf_v(即 cpf_v 将 $v\in s$ 的值从 d_1 转换到 d_2),后继状态 s'的启发值由转换代价 $\mathrm{cost}_v(d_1,d_2)$ 和 s'的报酬值两部分组成。$\mathrm{cost}_v(d_1,d_2)$可由算法 2 得到。s'的立即回报值可通过 reward 函数来计算。此外,如果 s'出现在一个状态轨迹中,还要乘上一定的折扣值。最后,状态的启发值是这两部分的加权组合。公式如下:

$$\mathrm{hval}=w_1\times\mathrm{cost}_v(d_1,d_2)+w_2\times R(s'),v\in s,d_1\xrightarrow{\mathrm{cpf}_v}d_2 \tag{5}$$

算法 3　$\mathrm{CGH}_{\mathrm{RDDL}}$算法

输入:待估值状态 s、固定轮数 h_{fix}。

输出:启发值 hval_s。

a)初始化。$h=1,\mathrm{hval}_s=0$。

b)状态演化至固定轮数 h_{fix}。

(a)在 CG 上随机取一条包含 action_fluent 的 cpf 应用到 s 中,所得某个 v 的值从 d 变化到 d',产生后继状态 s';

(b)$\mathrm{hval}_s+=(w_1\times\mathrm{cost}_v(d,d')+w_2\times R(s'))\times h'$;

(c)h 递增;

c)返回 hval_s。

算法 3 中,h_{fix}是一个可调参数。由于没有固定目标状态,所以启发值中的状态演化也必须设置一个固定轮数 h_{fix}。在实现中,为了加快启发值的计算速度,可采取忽略动作删除效果的类似策略。当不考虑动作删除效果时,状态演化可能会到达一个不动点,即应用任何动作都不会产生新的变化,这时无须再应用动作,只需计算不动点状态的报酬值乘以一个折扣值即可。

3.3　算法复杂性分析

算法 2 的时间复杂性分析如下。算法 2 基本上是一个最短路径算法。这是因为,如果存在边 $d_1\rightarrow d_2$,则 $\mathrm{cost}_v(d_1,d_2)$是边上的权值,且此权值为正。

算法 3 的时间复杂性分析如下。在算法 3 中,第 2 步状态演化的次数最多为 h_{fix}。每次状态演化,需遍历一遍 CG $=\langle V_{\mathrm{CG}},E_{\mathrm{CG}}\rangle$找可应用于当前状态 s 的 cpf。根据图搜索算法,这样的遍历耗时 $O(|V_{\mathrm{CG}}|+|E_{\mathrm{CG}}|)$。然后,根据该 cpf 作用的流文字 v,在 DTG_v 上找到 d_2(d_1 在 s 中已知)。设最复杂的 DTG 为 $\mathrm{DTG}_{\max}$,则找 d_2 耗时 $O(|V_{\mathrm{DTGmax}}|+|E_{\mathrm{DTGmax}}|)$。所以算法 3 的时间复杂性为 $O(h_{\mathrm{fix}}\times(|V_{\mathrm{CG}}|+|E_{\mathrm{CG}}|+|V_{\mathrm{DTGmax}}|+|E_{\mathrm{DTGmax}}|))$。如果是 h_{fix}固定不变的,则算法 3 相对于针对领域的 CG 和 DTG 是线性的。

算法 1 的时间复杂性分析如下。

第 1.1 步:耗时 $O(|V_{\mathrm{CG}}|+|E_{\mathrm{CG}}|)$。

第 1.2 步:每个 cpf$\in$CPFs 定义了一个状态变量 v 的值转换,构建 DTG_v 耗时 $O(|V_{\mathrm{DTG}v}|+|E_{\mathrm{DTG}v}|)$。设$|\mathrm{CPFs}|=n_v$,且最复杂的 DTG 为 $\mathrm{DTG}_{\max}$,则构建所有 DTG 耗时 $O(n_v\times(|V_{\mathrm{DTGmax}}|+|E_{\mathrm{DTGmax}}|))$。

第 1.3 步:在有向图 $G=\langle V,E\rangle$中,计算任意两点之间的最短距离,可以使用基于动态规划的 Floyd-Warshall 算法,其时间复杂性为 $O(|V|^3)$。因此,此步耗时 $O(n_v(|V_{\mathrm{DTGmax}}|^3))$。

第 1 步总耗时 $O(|V_{\mathrm{CG}}|+|E_{\mathrm{CG}}|+n_v\times(|V_{\mathrm{DTGmax}}|+|E_{\mathrm{DTGmax}}|^3))$。

第 2 步耗时 $O(1)$。

第 3.1 步:状态演化的次数最多为 H。每次状态演化由步骤 3.1~3.8 来完成。第 3.2 步生成后继状态集合 S',大小最多为$|\mathrm{CPFs}|=n_v$。第 3.3 步调用算法 3 对每个后继状态 $s'\in S'$计算启发值,总耗时 $O(n_v\times h_{\mathrm{fix}}\times(|V_{\mathrm{CG}}|+|E_{\mathrm{CG}}|+|V_{\mathrm{DTGmax}}|+|E_{\mathrm{DTGmax}}|))$。第 3.4 步选择最佳后继状态,耗时 $O(n_v)$。第 3.5~3.8 步的计算耗时 $O(1)$,其中调用 rddlsim 的 evolve 方法可看成常数时间。因此,整个第 3 步总耗时 $O(H\times n_v\times h_{\mathrm{fix}}\times(|V_{\mathrm{CG}}|+|E_{\mathrm{CG}}|+|V_{\mathrm{DTGmax}}|+|E_{\mathrm{DTGmax}}|))$。

第 4 步耗时 $O(1)$。

综上,算法 1 的时间复杂性为 $O(H\times n_v\times h_{\mathrm{fix}}\times(|V_{\mathrm{CG}}|+|E_{\mathrm{CG}}|+|V_{\mathrm{DTGmax}}|+|E_{\mathrm{DTGmax}}|)+n_v\times(|V_{\mathrm{DTGmax}}|^3))$。当 H 和 h_{fix}均为常数时,算法 1 的时间复杂性是 CG 和 DTG 这两个图的规模的多项式函数。

4　实验与分析

4.1　实验设置

本文提出的所有算法用 Java 语言来实现。其中,rddlsim 是 IPPC 提供的模拟器。实验内容分成两部分:

a)将本文方法与 PROST 和 Glutton 进行比较。本文方法基于因果图启发,PROST 基于模拟抽样,Glutton 基于迭代深化。此外,本文方法与 PROST 和 Glutton 的不同之处还在于后两者均没有使用 rddlsim,以至它们均不能对 RDDL 描述进行完全的解析。这一部分的实验内容是三种求解 PPP 问题的方法之间的性能比较。

b)将本文方法中的核心估值算法 $\mathrm{CGH}_{\mathrm{RDDL}}$与爬山法、随机搜索算法进行比较。这是因为 rddlsim 的估值函数是可配置的,可设计成不同的启发式算法。为了验证因果图启发的优越性,本文对 rddlsim 自带的爬山策略和随机搜索策略稍加修改,实现了不同的启发式策略来进行对比。其中,爬山法和随机法都通过应用 CPF 产生所有可能的后继状态。但是与 $\mathrm{CGH}_{\mathrm{RDDL}}$不同的是,爬山法仅计算后继状态的当前回报值来作为启发

值,不在因果图上进行任何状态演化;而随机法则不计算启发值,在后继集合中随机地选取一个作为后继状态。

实验的基准领域选取 IPPC 2011 的所有 MDP 领域(共八个),每个领域包含 10 个实例问题。PROST 和 Glutton 的运行环境为:Linux Ubuntu 12.04 + CPU 2.27 Hz(i3 350) + 内存 2 GB。本文方法的运行环境为:Windows 10 + CPU 2.40 GHz (i7-5500U) + 内存 8 GB。对全部基准领域的每个实例分别运行多次,取总和或平均的运行时间和奖赏值。全程无手工干预或参数调整。

4.2 实验结果分析

本文在所有实例上都运行了以上所说的规划器,包括 PROST、Glutton、CGH_{RDDL}(rddlsim + CGH)、RD_{RDDL}(rddlsim + Random)和 HC_{RDDL}(rddlsim + HillClimb)。后三者都使用 rddlsim 进行 RDDL 描述解析和状态演进,区别在于启发式不同。由于篇幅的限制,本文没有将每个领域的实验结果一一展示。表 1 给出了在 Elevator 领域三种求解方法 CGH_{RDDL}、PROST 和 Glutton 的结果对比。表 2 给出了在 Elevator 领域三种启发式 CGH_{RDDL}、RD_{RDDL}和 HC_{RDDL}的结果对比。

表 1 Elevator 领域的实验结果对比(CGH_{RDDL}/ PROST/ Glutton)

No	CGH_{RDDL}(R/T(ms))		PROST(R/T(s))		Glutton(R/T(s))	
1	-870	4 057	-138 5	1 308	-1 424.8	7.6
2	-536	8 072	-594.6	1 297	-646.4	32.5
3	-689	8 355	-168 2	1 344	-1 613.8	34.7
4	-958	1 751	-165 2	1 326	-1 627.4	10.4
5	-1 180	10 620	-155 5	1 318	-1 866.8	107
6	-1 298	7 097	-216 1	1 317	-2 375.7	105
7	-1 201	1 530	-246 8	1 371	-2 481.8	15.8
8	-1 387	8 372	-199 4	1 342		
9	-1 425	8 900	-285 6	1 364		
10	-1 080	1 807	-214 3	1 363		

表 2 Elevator 领域的实验结果对比(CGH_{RDDL}/RD_{RDDL}/ HC_{RDDL})

No	CGH_{RDDL}(R/T(ms))		RD_{RDDL}(R/T(ms))		HC_{RDDL}(R/T(ms))	
1	-87.6	310	-108.5	261	-87.6	304.6
2	-43.3	337	-100.4	239.3	-43.3	363.6
3	-65.8	424	-130.3	190.3	-74	295.3
4	-113	58	-150.6	32.6	-113	36
5	-113	390	-161.7	192	-113	203.6
6	-123.6	328	-152.3	128.6	-123	193
7	-110.3	26.6	-152.3	26.6	-110	55.3
8	-132.6	240	-199.8	149	-133	376.6
9	-125	230	-180.3	108.3	-160	233
10	-108	34.6	-117.8	22.3	-108	41.3

图 6 给出了 CGH_{RDDL}、RD_{RDDL}和 HC_{RDDL}在八个领域的平均回报值对比(横轴表示领域名,纵轴表示回报值)。图 7 给出了 CGH_{RDDL}、RD_{RDDL}和 HC_{RDDL}在八个领域的平均求解时间对比。图中横轴表示领域名,纵轴表示时间(ms)。其中要注意的是,为了方便与 PROST 和 Glutton 比较,表 1 中 CGH_{RDDL}对每个实例运行 20 次,取总和 reward 总的运行时间。而表 2 中,三种启发式对每个实例运行三次,取平均 reward 和平均运行时间。

对于实验内容的第一部分,表 1 表明在 elevator 领域,CGH_{RDDL}的求解性能比 PROST 和 Glutton 要好很多。其中,R 表示回报值,T 表示时间(ms 表示毫秒,s 表示秒)。由于全程无手工干预或参数调整,PROST 和 Glutton 的实际性能要比竞赛公布的结果差很多。在其他七个领域,比较结果也是类似的。

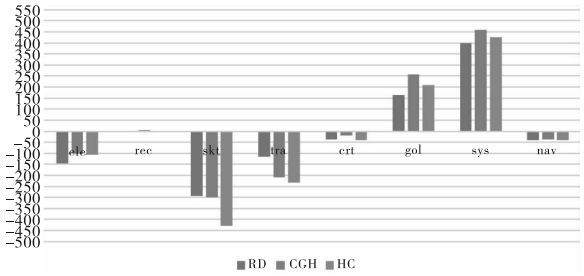

图6 CGH_{RDDL}/RD_{RDDL}/HC_{RDDL}在八个领域的平均回报值对比

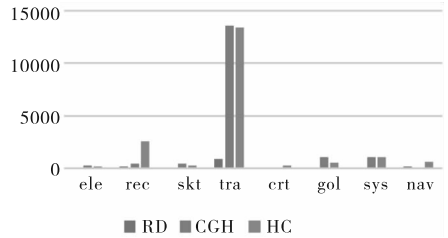

图7 CGH_{RDDL}/RD_{RDDL}/HC_{RDDL}在八个领域的平均时间对比

对于实验内容的第二部分,图 6 和 7 表明,对于大部分领域,CGH_{RDDL}的求解质量比 RD_{RDDL}要好,求解速度比 HC_{RDDL}快。求解质量用平均回报值来表示,求解速度用平均求解时间来表示。对于回报值而言,如果是负值,则绝对值越小越好;如果是正值,则绝对值越大越好。RD_{RDDL}由于采用随机策略,无估值代价,所以速度快但质量差。而 HC_{RDDL}要达到与 CGH_{RDDL}同样好的规划质量,则需付出更多的时间。图 8 出现了一些异常结果,即 CGH_{RDDL}和 HC_{RDDL}在 traffic 领域的求解时间很长,这可能是因为后继状态激增而导致估值时间过长的缘故。

5 结束语

并行不确定规划(PPP)是非常实用的规划模型,它结合了并行规划和概率规划的双重特征。然而现有的规划器在不允许手工干预和参数调整的前提下,求解效果并不理想。因此,本文将经典的因果图启发式方法引入到 PPP 问题的求解中,形成了高效且通用的针对 PPP 问题的启发式搜索算法。整体算法流程如下:a)在预处理阶段,根据 RDDL 领域描述构建因果图(CG)和领域转换图(DTG),然后在 DTG 上计算每个状态变量两个取值之间的转换代价;b)在求解阶段,使用 rddlsim 来模拟状态演化,每一步演化需采用由启发式估值算法评估的最佳转换,直至到达限定的轮数。核心的启发式估值算法称为 CGH_{RDDL}算法。其在 CG 上演化状态并计算状态轨迹的启发值。其中,单个状态的启发值定义为应用条件概率函数(CPF)的转换代价以及状态的回报值之和,而状态轨迹的启发值则为在轨迹上的所有状态启发值之和。因此,启发式求解 PPP 问题的目的在于,在状态演化的每一步,使用 CGH_{RDDL}算法为 rddlsim 推送最佳转换函数,从而使得状态演化的总回报值尽可能大。实验结果表明,本文所提出方法的求解效果好于两个求解 PPP 问题的代表性规划器,并且基于因果图的 CGH_{RDDL}算法的求解质量远高于随机搜索算法,而在保证相同求解质量的前提下,CGH_{RDDL}的求解速度远快于爬山法。

本文所提出的方法高效且通用,在求解过程中无须手工干预或参数调整。但是方法仍存在着一些不足,在未来的工作中

需要改进。首先,CG 和 DTG 的构建不能完全自动化。这是因为 CPF 的语法非常复杂,例如可以包含多层嵌套的条件表达式或概率分布函数,有些情况下需要人工解析或者使用额外的数据结构来标注。然而这并非本文方法独有的缺点,因为在 Fast Downward 中,这两类图的创建也是手工完成的。但是如何从 RDDL 描述中自动构建 CG 和 DTG,是一个非常值得研究的子课题。其次,在 CGH_{RDDL}算法中,固定轮数 h_{fix}是一个可调参数,其值目前本文设置为最大轮数 H 的 1/2 或者 1/3。h_{fix}的意义在于,由于 PPP 没有固定的目标状态,所以状态的演化无论在求解模拟中还是在估值中必须设置固定的步数。与其他的启发步长一样,h_{fix}的设置显然会影响启发值计算的效果,太大则耗费时间,太小则不够精准。在未来工作中,可研究如何设置一个可变的 h_{fix}值,使得其随着估值效果自适应地调整。最后,CGH_{RDDL}算法目前只能用于 RDDL 描述的 MDP 版本,需进一步将其扩展到 RDDL 描述的 POMDP 版本。这会涉及到对可观测流文字(observ-fluent)的特殊处理。

参考文献:

[1] 饶东宁,李建华,蒋志华,等. 并行概率规划综述[J]. 计算机应用研究,2016,33(6):1607-1611.

[2] Ghooshchi N, Namazi M, Newton M, *et al*. Transition constraints for parallel planning[C]//Proc of the 29th AAAI Conference on Artificial Intelligence. Palo Alto, CA: AAAI Press, 2015: 3268-3274.

[3] Vianna L, Barros L, Sanner S. Real-time symbolic dynamic programming[C]//Proc of 29th AAAI Conference on Artificial Intelligence. Palo Alto, CA: AAAI Press, 2015: 3402-3408.

[4] Sanner S. Relational dynamic influence diagram language (RDDL): language description [EB/OL]. (2011). http://users. cecs. anu. edu. au/ ~ ssanner/IPPC2011/RDDL. pdf.

[5] Keller T, Helmert M. Trial-based heuristic tree search for finite horizon MDPs[C]//Proc of the 23rd International Conference on Automated Planning and Scheduling. Palo Alto, CA: AAAI Press, 2013: 135-143.

[6] Kolobov A, Dai peng, Mausam D S, *et al*. Reverse iterative deepening for finite-horizon MDPs with large branching factors[C]//Proc of the 22nd International Conference on Automated Planning and Scheduling. Palo Alto, CA: AAAI Press, 2012: 146-154.

[7] Srinivasan S, Talvitie E, Bowling M. Improving exploration in UCT using local manifolds[C]//Proc of the 29th AAAI Conference on Artificial Intelligence. Palo Alto, CA: AAAI Press, 2015: 3386-3392.

[8] Bonet B, Geffer H. Labeled RTDP: improving the convergence of real-time dynamic programming[C]//Proc of the 13th International Conference on Automated Planning and Scheduling. Palo Alto, CA: AAAI Press, 2003: 12-21.

[9] Ghallab M, Nau D, Traverso P. 自动规划——理论和实践[M]. 姜云飞,杨强,凌应标,译. 北京:清华大学出版社,2008.

[10] Geffner H, Bonet B. A concise introduction to models and methods for automated planning [M]//Synthesis Lectures on Artificial Intelligence and Machine Learning. Williston, VT: Morgan & Claypool, 2013: 1-141.

[11] Hoffmann J, Nebel B. The FF planning system: fast plan generation through heuristic search [J]. Journal of Artificial Intelligence Research, 2001, 14(1): 253-302.

[12] Bonet B, Geffner H. Planning as heuristic search[J]. Artificial Intelligence, 2001, 129(1-2): 5-33.

[13] Helmert M. The fast downward planning system[J]. Journal of Artificial Intelligence Research, 2006, 26(1): 191-246.

[14] Helmert M, Geffner H. Unifying the causal graph and additive heuristics[C]//Proc of the 18th International Conference on Automated Planning and Scheduling. 2008: 140-147.

[15] Rao Dongning, Jiang Zhihua. Learning planning domain descriptions in RDDL [J]. International Journal on Artificial Intelligence Tools, 2015, 24(3): 1550002.

[16] Piacentini C, Alimisis V, Fox M, *et al*. An extension of metric temporal planning with application to AC voltage control [J]. Artificial Intelligence, 2015, 229(11): 210-245.

[17] Núñez S, Borrajo D, López C L. Automatic construction of optimal static sequential portfolios for AI planning and beyond [J]. Artificial Intelligence, 2015, 226(9): 75-101.

[18] Gerevini A, Long D. Plan constraints and preferences in PDDL3: the language of the fifth international planning competition, R. T. 2005-08-47 [R]. Italy: University of Brescia, 2005.

[19] Younes H L S, Littman M L. PPDDL 1. 0: an extension to PDDL for expressing planning domains with probabilistic effects, CMU-CS-04-167[R]. Pittsburgh, PA: Carnegie Mellon University, 2004.

[20] Manfredotti C V, Messina E V. Relational dynamic Bayesian networks to improve multi-target tracking[C]//Proc of the 11th International Conference on Advanced Concepts for Intelligent Vision Systems. Berlin: Springer-Verlag, 2009: 528-539.

[21] 饶东宁,蒋志华,姜云飞,等. 在部分观测环境下学习规划领域的派生谓词规则[J]. 计算机学报,2015,38(7):1372-1385.

(上接第 1349 页)

[11] 周象贤,金志成. 国外对平面广告受众注意心理的眼动研究[J]. 心理科学进展,2006,14(2):287-293.

[12] Eickhoff C, Dungs S, Tran V. An eye-tracking study of query reformulation[C]//Proc of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2015: 13-22.

[13] Liu Zeyang, Liu Yiqun, Zhou Ke, *et al*. Influence of vertical result in Web search examination[C]//Proc of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2015: 193-202.

[14] Chen Ye, Liu Yiqun, Zhou Ke, *et al*. Does vertical bring more satisfaction? Predicting search satisfaction in a heterogeneous environment [C]//Proc of the 24th ACM International Conference on Information and Knowledge Management. New York: ACM Press, 2015: 1581-1590.

[15] Liu Yiqun, Chen Ye, Sun Jiashen, *et al*. Different users, different opinions: predicting search satisfaction with mouse movement information[C]//Proc of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2015: 493-502.

[16] 曹卫真,车笑琼,祁禄,等. 画面主体位置布局的眼动实验及对网络视频资源建设的启示[J]. 远程教育杂志,2013(5):97-106.

[17] 苌道方,钟悦. 考虑行为和眼动跟踪的用户兴趣模型[J]. 河南科技大学学报:自然科学版,2014,35(1):49-52.

[18] 施卓敏,郑婉怡. 探秘不同认知风格的个体关注广告的差异——广告位置和认知风格对广告效果影响的眼动研究[J]. 营销科学学报,2014,10(3):128-145.

[19] 柴林麟. 大数据时代下互联网广告及计算广告学的应用研究[J]. 信息与电脑:理论版,2015(16):29-31,56.

[20] 周傲英,周敏奇,宫学庆. 计算广告:以数据为核心的 Web 综合应用[J]. 计算机学报,2011,34(10):1805-1819.