软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2013,24(5):1061−1077 [doi: 10.3724/SP.J.1001.2013.04264]

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

# PDDL 的 ER 建模以及开发基于存储过程的规划器*

饶东宁[1], 蒋志华[2], 姜云飞[3]

[1](广东工业大学 计算机学院,广东 广州 510006)

[2](暨南大学 信息科学技术学院 计算机科学系,广东 广州 510632)

[3](中山大学 信息科学技术学院 软件研究所,广东 广州 510275)

通讯作者: 蒋志华, E-mail: tjiangzhh@jnu.edu.cn

**摘 要**: 随着智能规划的发展,其所面对的问题规模越来越大,而且可以预见以后会更大.现有的研究大多用二级存储扩展空间,其终极形式应该是用数据库进行存储.此外,有很多同一领域的规划问题,其所包含的常量几乎一致,其中必然有可重用信息来帮助加速求解.要更好地利用这些可重用信息也需要数据库.考虑到以上两个问题,首次提出规划领域描述语言 PDDL(planning domain description language)的 ER 模型(entity relationship model),并基于此模型用存储过程来编写规划器 SPP(stored procedure planner).SPP 是完全在数据库内部运行的最优规划器,存取效率高,可充分利用数据库的各种功能.在国际规划大赛 IPC(Int'l planning competition)基准领域上的实验结果表明,在有限的机器配置下,SPP 可以求解传统最优规划器不能求解的问题.该工作迈出了在数据库中求解规划问题,从而彻底解决空间问题的第一步.

**关键词**: 人工智能;智能规划;规划系统;存储过程;关系数据库管理系统

**中图法分类号**: TP181 **文献标识码**: A

中文引用格式: 饶东宁,蒋志华,姜云飞.PDDL 的 ER 建模以及开发基于存储过程的规划器.软件学报,2013,24(5):1061−1077. http://www.jos.org.cn/1000-9825/4264.htm

英文引用格式: Rao DN, Jiang ZH, Jiang YF. Building entity relationship models for PDDL and developing planners based on stored procedures. Ruan Jian Xue Bao/Journal of Software, 2013,24(5):1061−1077 (in Chinese). http://www.jos.org.cn/1000-9825/4264.htm

## Building Entity Relationship Models for PDDL and Developing Planners Based on Stored Procedures

RAO Dong-Ning[1], JIANG Zhi-Hua[2], JIANG Yun-Fei[3]

[1](School of Computer, Guangdong University of Technology, Guangzhou 510006, China)

[2](Department of Computer Science, Jinan University, Guangzhou 510632, China)

[3](Software Research Institute, School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510275, China)

Corresponding author: JIANG Zhi-Hua, E-mail: tjiangzhh@jnu.edu.cn

**Abstract**: With the development of automated planning, the size of problems is getting bigger and bigger, and one can predict that it will become very large in the future. Some existing research work begins to use secondary memories to extend the search space, and it is believed databases are finally used. Besides, many problems that belong to the same domain often have common constants, so there might be a plenty of reusable information to speed up the solution process. To store this information permanently, databases are also needed. Inspired by the above two reasons, this paper first proposes ER (entity relationship) models for PDDL (planning domain description languages) and then develops a stored procedure based automated planner (stored procedure planner, SPP) for the first time. This planner is an optimal one which runs totally inside a database. It stores and accesses data efficiently and takes fully advantages of database

* 基金项目: 国家自然科学基金(61003179, 61100134); 广东省自然科学基金(S2011040001427)
收稿时间: 2011-04-05; 修改时间: 2012-03-19; 定稿时间: 2012-05-18

features. Experiments on benchmark problems from the Int'l planning competition (IPC) show that this planner can solve problems which cannot be solved by some classical optimal planners in a limited machine configuration. The work in this paper takes the first step for solving planning problems totally in a database so that it is helpful to solve huge-size problems finally.

**Key words**: artificial intelligence (AI); automated planning; planning system; stored procedure; relational database management system (RDBMS)

智能规划是人工智能的重要分支,近 20 年来取得了很大的进展,其处理的问题规模也越来越大.人工智能领域顶级期刊 Journal of Artificial Intelligence 分别在 1995 年、2003 年、2009 年出版了智能规划与调度的专刊,可见其重要性.从 1998 年开始的两年一度的国际规划大赛 IPC(Int'l planning competition)也吸引了越来越多的关注,并多方面地推动了规划的发展.其中,不容忽视的一点是问题的规模越来越大.比如,在 2000 年的 IPC-2 (http://www.cs.toronto.edu/aips2000/,即第 2 届 IPC,以下标记方法相同)上,Logistics 领域规模最大的问题(probLOGISTICS-15-1.pddl)只有 5 个 location 对象,而在 2008 年的 IPC-6(http://ipc.informatik.uni-freiburg.de/planners)中,类似的 Transport 领域中最大的问题(p30.pddl)有 60 个 location 对象.考虑到实际问题中的对象可能更多(如例 1 所示),未来智能规划要解决的问题规模还会进一步扩大.

例 1:与 IPC-3 中 Zeno travel 领域(http://planning.cis.strath.ac.uk/competition/domains/zenotravel.tgz)相似的民航客运问题在实际生活中涉及到民航机场、飞机等对象,其中,到"十一五"末,中国民航运输机场总数将达到 190 个左右,而 Zeno travel 领域的基准测试中一般只有 20 多个与机场对应的 airport 对象.也就是说,现实问题 10 倍于规划大赛基准用例中的问题规模.

大规模问题带来了巨大的搜索空间.目前已有在搜索中利用二级存储的技术来解决内存不足的问题[1−9],而最终会发展到利用数据库来进行存储.此外,有很多问题不仅属于同一领域,而且所涉及的常量也基本相同.在最新的规划定义语言 RDDL(relational dynamic influence diagram language,http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf)中,整个描述被分成 3 部分:domain,non-fluent,instance.其中,non-fluent 的提出正是基于上述原因.当然,出于信息重用的考虑也需要利用数据库.如例 1 中,旅客的出发地和目的地变更就变成了另一规划问题,但是其总数,甚至包括机场和飞机的总数,都可以是相同的.假如存储了规划过程中的信息,那么重用已有的搜索信息就可能更快地找出规划解.

鉴于上述两个方面的原因,本文提出规划领域描述语言 PDDL(planning domain description language)的 ER (entity relationship)模型,并基于此模型用存储过程来编写规划器 SPP(stored procedure planner).ER 模型刻画了 PDDL 中的全局实体、领域实体和问题实体以及它们之间的关系.SPP 完全运行在数据库内,相对于客户端/服务器等其他方式而言数据存取效率更高.此外,由于采用可采纳(admissible)的 $A^*$算法,SPP 找到的解是最优解.得益于数据库巨大的存储空间,在同样的机器配置下,它可以求解其他最优规划器[10]不能求解的问题.SPP 以基于单个目标可达性为启发式函数,对搜索中的剪枝和已有信息的利用进行了特殊处理,即在基于搜索的经典规划器 HSP[11]的思想和技术上进行了改进.尽管现阶段 SPP 只能处理 STRIPS 类型的[12]规划问题,但是其对更多语言特性的扩展预留了空间.这是首次数据库被用于求解规划问题的独立平台,而不是仅仅作为信息存储或知识推理的辅助工具[13].这也是首次尝试使用存储过程编写规划器,而不是基于通用的程序设计语言.

本文第 1 节介绍本文的研究背景,包括外存规划的发展和规划中常用的启发式算法.第 2 节介绍 SPP 的数据结构,即 PDDL 的 ER 模型.第 3 节介绍 SPP 的主要算法,包括搜索算法、启发算法以及利用 SQL 语句设计的特殊算法.第 4 节介绍本文的实验.第 5 节是结语以及对未来工作的展望.

## 1 研究背景

智能规划领域研究计算机系统在执行任务之前如何自动制定行动步骤,其研究成果近年来得到了整个人工智能领域的认可.简单地说,一个规划问题主要由初始状态描述 $I$、目标状态描述 $G$ 和动作集 $A$ 组成[2].规划的过程就是寻找从 $I$ 到达 $G$ 的一个动作序列.

### 1.1 基于外存的规划技术

魔方、汉诺塔等问题的最优解往往需要搜索百万的三次方的状态空间.即使对于启发式搜索来说,这样的空间也太大了,在极端的情况下,更需要 TB 级的存储器.显然,这个时候仅仅使用内存是不够的,需要借助于外存.1995 年,Chiang 等人提出外存图规划[1];2000 年,Krof 和张维雄等人提出利用分治改进 A*算法[2]以利用外存进行搜索;2004 年,Edelkamp 等人进一步提出了外存 A*算法[3].

在这些外存搜索算法中,宽度优先搜索(breadth-first search,简称 BFS)是最重要的一种.2002 年,Mehlhorn 和 Meyer 提出了 MM_BFS[4].它包括一个预处理程序来重构图的邻接表,按距离聚簇.其思想是:当同一个簇中第 1 个节点被访问时,该簇的其他节点也很可能马上被访问.后来的很多研究讨论了在更大的图上进行 BFS 的算法,但仍然都基于 MM_BFS 算法.Zhou 和 Hansen 在 2006[5]年提出了利用分支限界法来改进 BFS 算法,其中特别改进了内存冗余处理方法,只存储搜索最前沿的节点.他们展示了 BFS 算法可以比最佳优先策略有更好的效果.Idwan 在 2009 年[6]提出一种在大图上进行 BFS 的优化算法.它将整个图划分为小图,每个小图的节点数一致而且小图之间的边数最小.每次在一个子图上进行 BFS,然后利用子图之间的边选择下一个子图,直到所有的节点被访问为止.

随着规划问题规模的扩大,外存规划(external memory planning)也在 2006 年[7]被提出来.其中利用了外存 BFS,而且利用访问的局部性避免重复访问.为了处理马尔可夫决策过程(Marcov decision processes,简称 MDP)规划的可扩展性问题,Edelkamp 等人另外提出了一个值迭代算法的变形,称为外存值迭代[8].该算法是第一个利用外存来存储 MDP 模型,其 MDP 模型的执行和备份是在内存和外存之间进行交换的.为了解决随机的存储顺序带来的低效问题,他们在每次迭代之前对转换函数进行排序.2008 年,Dai 等人[9]对此提出的解决方案是将 MDP 分成小块,只把相关的块放在内存中,然后一块块地进行更新.

与一般的算法不同,在外存算法中需要具备很多基础设施,如虚拟内存管理[3]和磁盘运作方式[4]等.进一步地,移除重复节点[5]以避免重复扩展也很关键,不断地进行外部排序[4]也是必要的.实际上,假设问题足够大,那么索引、压缩等功能也是必须的,这时的外存算法几乎包括了数据库的所有主要功能.因此,不但从存储空间上来看利用数据库是外存算法的必然趋势,而且从功能上看也是如此.

### 1.2 规划中的启发式搜索技术

最早的规划求解往往是通过鲁宾逊归结原理求解联合子目标或者利用情景演算等方法进行推理.其后,人们提出了利用启发式进行状态空间或者规划空间搜索、转换为 CSP 或者 SAT 等问题来进行求解等.IPC 对于验证这些方法的效用起到了重要的作用.在 IPC-1 上,除 HSP[11]外的其余规划器都是基于图规划算法的,IPC-2 最后评选出两个最优秀的规划系统,分别是基于部分序规划的 STAN 和基于启发式搜索的 FF[14].之后,FF 仍然在 IPC-3 的 Strips 层次上性能最优.IPC-4 和 IPC-5 提高了对规划解质量的要求,最优规划器 SATPlan[10]的后继版本均获得了优胜.可见,在不要求最优解的情况下启发式方法占优,在要求最优解的情况下借助 SAT 技术求解的方法占优.但是,SAT 方法对空间的要求非常大而且难以利用外存帮助快速求解,于是,SATPlan[10]后继版本的作者之一 Edelkamp 在 2006 年[7]提出用外存 BFS 进行规划.在 IPC-6 上,问题规模进一步扩大,基于外存的算法在竞赛这种强调速度的情况下劣势明显,改进了 FF 中启发式算法的 Lama[15]获得了优胜.

主要的可采纳启发式包括:基于放松式规划图的启发式 $h^+$[14]、关键路径启发式 $h^m$[16]以及结构化模式[17]等抽象启发式方法.$h^+$忽略动作的删除效果,将由此得到的放松式规划图中的距离作为目标可达程度的估计,SATPlan[10]的后继版本也采用这种技术进行可达性分析.对 HSP[11]的启发式方法的进一步分析带来了 $h^m$,其核心思想是将从当前状态到目标状态的一个子集($m$ 个目标)的距离作为启发式估计.抽象启发式方法[17]大多采用抽象命题空间的表示方式来简化动作和状态模型,进而得到简化的规划解长度作为启发值.

其中,在重用搜索信息方面,$h^1$(即在 $h^m$ 中,令 $m$=1)[16]最实用.首先,$h^m$ 比 $h^+$具有明显的优势,因为它计算的对象是真实状态[16],而不是抽象状态[17]或者放松动作[14]的结果.进一步来说,由于考虑到计算代价,并且每个原子目标的关键路径对于任意状态距离估计都是必需的,$h^1$ 比 $h^m$ 更容易实现,因而也更实用.

### 1.3 存储过程

存储过程(stored procedure)是经编译后存储在数据库中的 SQL 语句集.其优点包括:可保证数据的安全性和完整性;数据库可对其进行深度优化使其执行速度快于直接执行动态SQL语句;可以降低网络的通信量;可以集中控制业务规则.其缺点主要体现在调试困难和移植困难等方面.

正是由于存储过程的这些优点,主流数据库包括 Oracle,Sybase,SQL Server 等很早就开始支持这种语言.随着开源数据库的发展,近年来,Mysql 等开源数据库也支持了这一技术.开源数据库对存储过程的支持对于学术研究有着其独特的意义.采用商业软件实现一些学术研究中需要的功能有两个方面的问题:首先无法判断其正确性和合理性,因为无法确知其实现方法;其次,也无法判断学术研究是否可延续,因为这样的系统不具有完整的版权.因此,Mysql 等开源数据库对存储过程的支持增加了在学术研究中使用存储过程的可行性.

## 2 PDDL 领域的 ER 模型

考虑到问题规模和信息重用的问题,本文提出了在数据库中求解规划问题的方法.进一步来说,如果使用数据库代替内存,那么最耗费时间的操作就在于数据存取.为了达到最快的存取速度,应该尽可能多地使用存储过程,以便数据库对此进行优化.此外,为了使用关系数据库,必须建立相关问题的 ER(entity relationship)模型.它是数据模型的典型代表,是用户和数据库设计人员之间进行交流的工具.其基本元素是实体、属性和关系.ER 模型到关系模型,特别是在关系数据库上进行相互转换,已经有了大量的工具支持.主流的大型数据库,包括 SQL Server 等都提供了从数据库提取 ER 模型的工具.而从 ER 模型构建数据库更有大量的工具,比如 Power Designer, ERWin 等.因此,在实践层面上,ER 模型、关系模型以及关系数据库是一致的.本节讨论从规划领域定义语言 PDDL 构建规划领域的 ER 模型.

### 2.1 PDDL语义

在 PDDL2.1[18]中指出,PDDL 的语义是基于由初始状态模型 $M_0$ 和动作描述 $o_p$ 构成的系统,其中,动作描述 $o_p$ 是一个包括前提、删除效果和增加效果的三元组.在集合论的层面上,PDDL 的语义模型定义了一组集合,而这些集合的元素之间语义相关.从关系模型的角度来看,集合的元素就是实体,每个实体都有自己的类型,实体之间的语义相关性就是关系.

其中,最基本的语义包括:

一个规划领域是个四元组,包括函数符号、关系符号、动作和函数参数映射,其中,函数符号简称函数,关系符号即谓词,动作是指动作模型.一个规划问题包括其所属的领域描述和初始逻辑状态以及最终逻辑状态.

**定义 1(原子集合(atoms))**. 原子集合 $Atms_I$ 是规划实例 $I$ 中关系符号对所有对象应用所得的实例,将这些关系符号视为一阶谓词时它就是所有命题的集合.

**定义 2(逻辑状态和状态(logical states and states))**. 给定一个有限原子集合 $Atms_I$,一个逻辑状态是 $Atms_I$ 的子集.一个状态$(t,s,x)$由时间 $t$、逻辑状态 $s$ 以及一个由规划实例的所有原始数字表达式的值构成的向量 $x$ 组成.

**定义 3(基本动作(ground actions))**. 给定一个规划实例,则动作模型 $A$ 的基本动作 $a$ 是将 $A$ 中参数实例化的结果集.集合中每个元素名称为动作模型的名称,命题前提称为前提 $Pre_a$,正文字效果称为增加效果 $ADD_a$,负文字效果称为删除效果 $Del_a$,数字化效果称为数字化效果 $NPE_a$.

**定义 4(动作可应用性(applicability of an action))**. 基本动作 $a$ 可应用意味着 $Pre_a$ 是逻辑状态 $s$ 的子集,也称 $Pre_a$ 被满足.

**定义 5(事件执行(happening execution))**. 给定一个状态$(t,s,x)$和一个事件 $H$,$H$ 的活动是一个基本动作 $a$,$a$ 的名字在 $H$ 中,并且可应用于$(t,s,x)$.其执行的结果是一个状态$(t',s',x')$,$s'=(s\backslash Del_a)\cup ADD_a$,而 $x'$ 是将 $NPE_a$ 应用于 $x$ 的结果.

**定义 6(简单规划(simple plans))**. 一个简单规划是一个时间-动作二元组的有限集合.

### 2.2 建模概述

PDDL 是描述规划领域和问题的标准语言,只有经过这样的形式化描述,才能被规划系统读入、解析、求解并输出规范的规划解.根据 PDDL 的语义及其 BNF 描述[18],本文设计了规划领域的 ER 模型,建模工具为 CA Erwin(http://erwin.com/).图 1 首先展示了规划领域中最重要的一些实体,包括命题、类型、状态和规划等.其中,双粗点实线表示多对多关系,单粗点实线表示识别关系,在工具中均有隐藏的表格来描述这些关系.关于实体的表示方式,首先要说明的是其命名规则.由于实体要么属于某个领域,要么属于领域的某个具体问题,因此在实体名前加$Domain 以区分不同领域的实体,加$omain_$Problem 以区分不同问题的实体.例如,类型是属于领域的,命名为$Domain_type;而常量一般是属于问题的,命名为$Domain_$Problem_constant.每个实体名下面的方框包括了实体的主要属性,其中,FK 表示外键值.其次,实体的建立是与 PDDL 的基本语义相一致的.例如在定义 2 中,一个状态$(t,s,x)$由时间 $t$、逻辑状态 $s$ 以及一个由规划实例的所有原始数字表达式的值构成的向量 $x$ 组成.对应到如图 1 所示的状态实体$Domain_$Problem_state,其包括 3 个对应的属性:时间实例 time、逻辑状态实例 logic_state_ID、PNE 值向量 PNE_vector_ID.

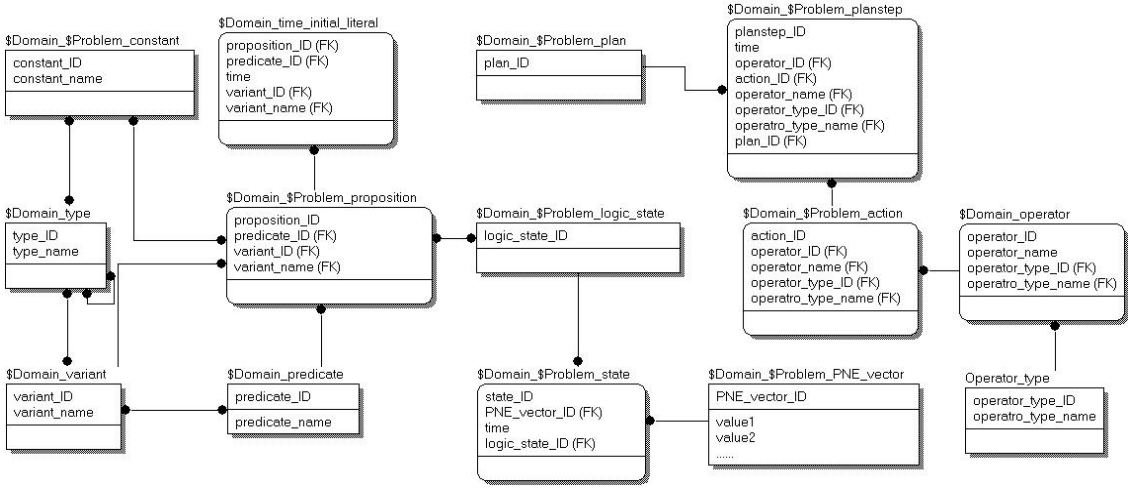


Fig.1 Entity relationship related to propositions, types, states and plans

图 1 命题、类型、状态和规划相关实体关系

然而,图 1 所示的部分实体过于笼统,应进行更细致的划分.例如,动作模型实体$Domain_$Problem_action 可进一步细分为一般动作模型、可持续动作模型和派生谓词规则,并且不同类型的动作模型具有不同的属性.分析可知,最基本的一类实体是在所有领域中都可以出现的,它们大多数是 PDDL 的保留字,例如运算符.另外,有些实体仅属于某一领域,例如类型;而另一些实体特属于某一问题,例如目标.因此,可根据实体的作用域(所属关系)来进行分类,PDDL 中的实体可分为全局实体、领域实体和问题实体这 3 类.

**定义 7(全局实体(global entities))**. 全局实体是作用域包括所有规划领域的实体.

限于篇幅,这里不再对全局实体及其属性值一一列举.总的来说,规划领域的全局实体包括运算符 operation、约束 constraint、最优化 optimization、量词 quantifier、需求 requirement、变量来源 variant_source 等等.其中,variant_source 不是 PDDL 的保留字,但是由于变量可以在动作模型或者谓词中出现,因此变量名不能唯一标识变量,需要结合变量的来源进行区分.类似地,还可定义领域实体和问题实体,分别见定义 8 和定义 9.

**定义 8(领域实体(domain entities))**. 邻域实体是作用域限定在某个规划领域的实体.

**定义 9(问题实体(problem entities))**. 问题实体是作用域限定在某个规划问题的实体.

图 2 展示了基本的领域实体及其属性,它们构成了规划领域描述的主要内容.例如,$Domain_requirement 表示领域需求,具体的属性值可包括:strips,:typing,:fluents,:adl 等,规划系统在正式求解之前必须判断自己是否具

有满足这些需求的能力.再如,$Domain_predicate 表示领域中所允许出现的谓词,这些谓词的实例化构成了问题的状态空间.在实际设计中,由于效果$Domain_effect、目标$Domain_goal、表达式$Domain_expression、动作模型$Domain_operator 具有不同数目和类型的属性(尽管它们的语法结构相似),为了避免大量空值,需要将它们视为不同的实体.

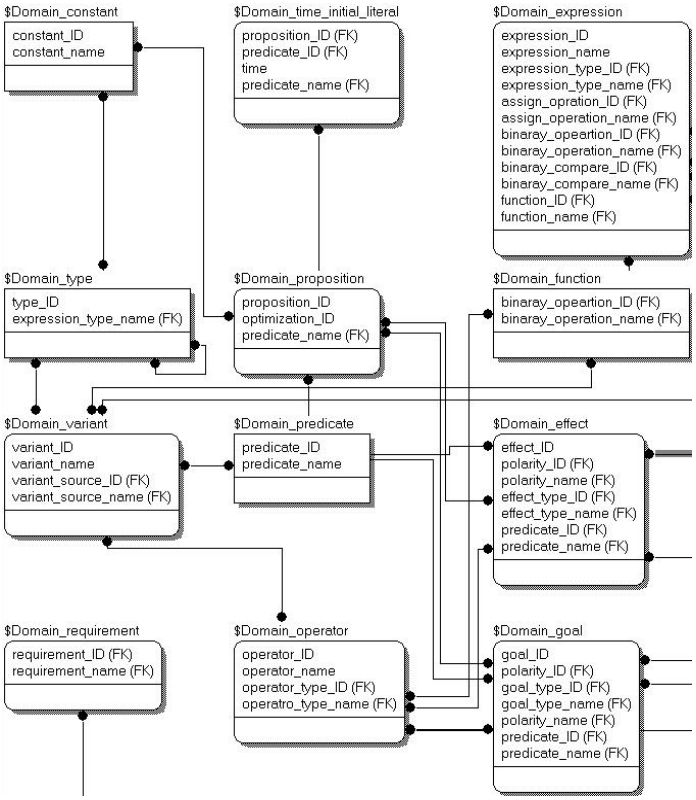


Fig.2 Overview on domain entities relationship

图 2 领域实体关系(概览)

图 3 是图 2 的另一种视图,它隐藏了实体的属性,而将实体之间的关系表显式地表示出来,这些表是由建模工具根据键值关系自动产生的.在图 3 中,直角矩形表示实体,圆角矩形表示关系表.例如,$Domain_constant_type 表示实体$Domain_constant 和$Domain_type 之间的关系表,将原来的多对多关系转换为识别关系.图 4 展示了部分问题实体及其关系.领域实体显然能够在问题中被使用,因此,它们中的大部分都需要被继承为问题实体.由于图 1 已给出与状态和规划相关的实体,因此图 4 仅展示了与问题直接相关的实体.例如,$Domain_$Problem_initial_state 和$Domain_$Problem_goal 分别表示规划问题描述中的两个主要元素,即初始状态和目标条件.

最后,给出关于建模的正确性讨论.从一种模型转换为另一种模型,原则上需要证明其转换过程的语义不变性.本文从 PDDL 的 BNF 描述中建立 ER 模型,面临同样的语义不变需求.但是,ER 模型的正确性往往通过实例的求解来说明[19,20].在本文第 4 节给出的实验部分也遵循这一习惯.从目前的研究情况来看,一方面,在数据库中 ER 模型的语义体现在数据的完整性和一致性(例如图 1~图 3 中的实体外键形成的完整性约束);另一方面,在规划领域中,将 PDDL 转换为其他形式系统往往也没有语义正确性方面的论证,例如转换为 CSP 或 SAT[10]等.再比如与本文工作有相关联系的建模和求解的集成工具 VLEPPO[21],它简单地将类型对应为实体,谓词对应于关系,使得规划领域和问题的 PDDL 描述可图形化地演绎为 ER 模型,并可调用底层规划系统进行求解.但是,其建立的 ER 模型并不能在数据库中进行求解,只是 PDDL 领域或问题的一种可视化的图形化描述,当然更不存在语义

一致性问题.因此,本文的建模正确性通过实例求解来进行验证,而不是从理论上进行分析.

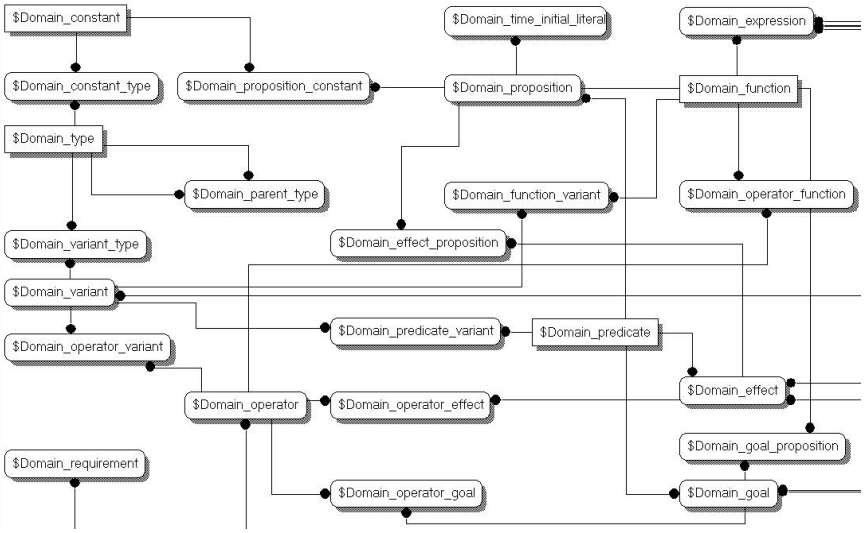


Fig.3 Explicit relationships between domain entities

图 3 领域实体的显式关系描述

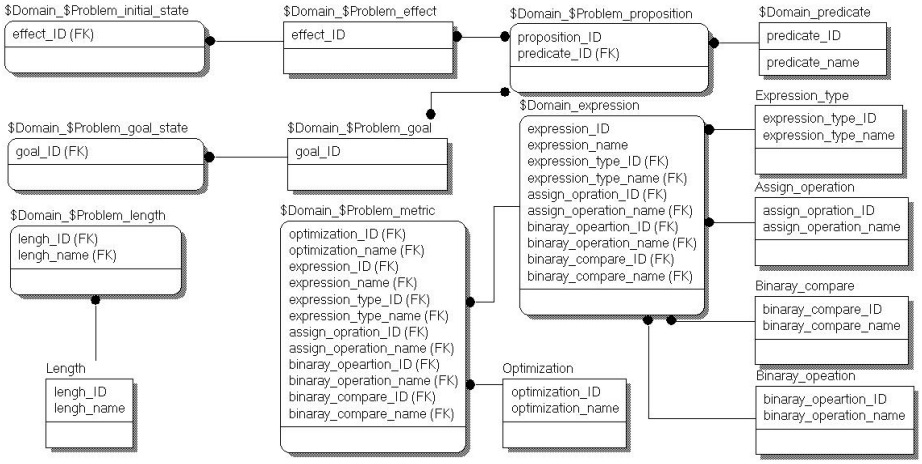


Fig.4 Problem entity relationship

图 4 问题实体关系

## 3 SPP

本节介绍基于存储过程的规划系统 SPP 的设计和实现.在通过 PDDL 语义建立了规划领域和问题的 ER 模型之后,可以利用存储过程直接编写规划系统进行求解.SPP 完全运行在数据库内,相对于客户端/服务器等其他方式而言数据存取效率更高.并且,由于使用可采纳启发式的 A*算法,SPP 能够返回最优解.下面首先介绍主要的搜索算法,然后介绍主要的启发函数和一个邻居发现算法,最后说明 SPP 实现的方式.

### 3.1 搜索算法

从根本上说,SPP 是基于 A*算法的.A*算法是最短路径搜索等问题的有效方法,其公式表示为

$$f(n)=g(n)+h(n), \tag{1}$$

其中,$f(n)$是从初始点经过节点 $n$ 到目标点的估价函数,$g(n)$是状态空间中从初始节点到节点 $n$ 的实际代价,$h(n)$是从节点 $n$ 到目标节点最佳路径的估计.能否找到最优解解,取决于估价函数 $h(n)$是否可采纳,即是否满足估价值 $h(n)\leqslant$($n$ 到目标节点的距离实际值).其搜索过程需要两个表:OPEN 表和 CLOSE 表.OPEN 表保存所有已生成而未考察的节点,CLOSED 表中记录已访问过的节点.其中,对 OPEN 表中的节点按照估价函数值排序,并依次进行扩展.如果扩展中涉及到 CLOSE 表中的节点,则需进一步考虑该节点是否需要移动到 OPEN 表.在扩展到目标节点后,A*算法通过一个回溯来找解路径,称为路径重构.

SPP 采用了关键路径启发式 $h^1$(详细介绍见第 2.2 节)作为启发函数.通过 $h^1$,SPP 计算从状态 $s$ 到每个原子目标 $g$ 的代价 $c(s,g)$,并以其中最大的那个 $c$ 作为 $s$ 到整个目标集 $G$ 的最小代价,即 $c(s,G)=\max_{g\in G}c(s,g)$.为了结合 $h^1$ 启发函数的特点,SPP 在 A*算法中嵌入了对启发值的处理.特别地,SPP 在求启发值过程加入了剪枝技术,即增加一个数据库表记录每个状态到每个原子目标的最短路径长度.如果在求值过程中当前路径超过了该长度就剪枝;否则,更新该值.做了上述调整后的 A*搜索算法如图 5 所示,其中,启发式的具体计算方法 *pruned_h*$^1$(*s*, *g*,*maxH*)如图 6 所示,找邻居状态的算法 *neighbor_nodes*(*x*)如图 7 所示.

```
Algorithm 1. Adapted A* algorithm.
1:   Input: $Domain, $Problem.
2:   Output: plan (as records in table).
3:     //initialization
         create necessary tables;
4:   maxH:=a large number;   //indicate maximum heuristic value
5:   closeset:=empty;
6:   openset:=initial-state node start in the problem;
7:   g[start]:=0
8:   //instead of finding neighbors, SPP calls computing h^1 at the beginning
      for each atom goal g do
9:       call pruned_h^1(start,g,maxH)
10:   h[start]:=the maximum heuristic value from start to atom goal
11:   //updating maximum heuristic value to reduce search
    if h[start]<maxH then maxH:=h[start]
12: //standard procedure of A* algorithm begins
    f[start]:=h[start]
13: while openset is not empty do
14: x:=the node in openset having the lowest f[] value
15:     //when goals are found, reconstruct plan path
    if x include the goal set of the problem then
16:       return reconstruct_path(came_from,x)
17: remove x from openset
18: add x to closedset
19:   //resetting maximum heuristic value
    maxH:=a large number;
20: for each y in neighbor_nodes(x) do
21:       if y in closedset then continue
22:       tentative_g:=g[x]+dist_between(x,y)
23:       if y not in openset then
24:           add y to openset
25:           tentative_is_better:=true
26:           elseif tentative_g<g[y] then
27:               tentative_is_better:=true
28:               else tentative_is_better:=false
29:       if tentative_is_better=true then
30:               came_from[y]:=x
31:               g[y]:=tentative_g
32:               for each atom goal g do
33:                   call pruned_h^1(y,g,maxH)
34:   //y should have the minimum distance estimation
                h[y]:=the minimum heuristic value
35:             if h[y]<maxH then maxH:=h[y]
36:             f[y]:=g[y]+h[y]
37: return failure
```

Fig.5 Adapted A* algorithm

图 5 修改的 A*算法

```
Algorithm 2. pruned_h^1 algorithm.
1:  Input: state s, atom goal g, pruned heuristic value maxH.
2:  Output: distance from s to g, that is c(s,g).
3:  //initialize the current depth
    h:=0;
4:  //check if existed
    if c(s,g) already exists in the table then return c(s,g);
5:  else if s includes g then return c(s,g):=0;
6:  //similar to the standard A* procedure
7:  while openset is not empty do
8:  //pruned cases
    if h>maxH then return c(s,g):=maxH+1;
9:  //increasing search depth
    h:=h+1;
10: x:=the node in openset having the lowest f value
11: //goal found, reconstruct plan path
    if x include g then
12:    return reconstruct_path(came_from,x,s)
13: for each y in neighbor_nodes(x) do
14:       if y in closedset then continue
15:       tentative_g:=g[x]+h;
16:       if c(y,g) exists then f[x]:=c(y,g)+h;
17:       else
18:          determine if updating is needed;
19:          if needed then
20:                came_from[y]:=x
21:                g[y]:=tentative_g
22:                call pruned_h^1(y,g,maxH–h)
23:                f[y]:=g[y]+h[y]
24: return failure
```

Fig.6 $h^1$ algorithm with pruning

图 6 带剪枝的 $h^1$ 算法

```
Algorithm 3. neighbor_nodes algorithm.
1:  Input: state s.
2:  Output: neighbor states.
3:  if neighbor states of s already exist
4:  then goto 11;
5:  for each operator o do
6:        //listing applicable actions
    SELECT action from (PREC INNER JOIN propositions in s) WHERE count(action)=precondition number of o
7:  for each applicable action a do
8:     //evolve state
    SELECT proposition into new state s′ from ((propositions in s UNION added effects of a) MINUS deleted effects of a)
9:     //if the successor state is new
    IF NOT EXISTS state ID from state table WHERE state magic number=
    (SELECT SUM(magic number of propositions in s′) from s′)
    THEN insert s′ to state table
10:    insert transition s —a→ s′ to state_action_state table
11: //get neighbor states
    SELECT to_state from state_action_state table where from_state=s
12: return neighbor states
```

Fig.7 Finding neighbors algorithm

图 7 邻居发现算法

在如图 5 所示的算法中,输入为表示领域名和问题名的变量,输出为规划解,以记录的形式出现在数据库表中.第 3 行~第 7 行为初始化表和变量.第 8 行~第 11 行是调用启发值的函数来计算起始节点到每个目标的 $h^1$ 值,以得到剪枝界限 *maxH* 更准确的初值.*maxH* 的作用在于剪枝,以节省搜索时间.第 12 行~第 37 行基本上是一个标准的 A*算法过程,其中,第 15 行、第 16 行在找到目标后进行路径重构;第 19 行将剪枝界限 *maxH* 重置以适应新的状态;第 20 行~第 36 行扩展当前节点,将未访问过的邻居节点放入 OPEN 表中,并以邻居节点到目标原子

的最小 $h^1$ 值作为新的剪枝界限.剪枝界限的调整得益于数据库表对已有信息的存储便利.

最后,对上述算法复杂性做初步的讨论.已有研究工作表明,由于规划问题是 NP 复杂的[22],仅有命题的规划问题是 NP 完全的[23],因此,目前来说不存在求解规划问题的多项式时间的高效算法.另外,A*算法的复杂性取决于搜索空间的大小和启发式函数的性能,此处采用 $h^1$ 作为启发函数,已经在最大程度上减少了启发值的计算代价.尽管 SPP 的复杂性仍然是 NP 的,但是其冗余计算一定是最少的:由于 SPP 基于数据库,所有计算结果都会被存储,避免冗余计算;进一步来说,由于数据库可对领域重用,在最大限度内避免了重复计算.具体的算法剪枝效果和时间性能可见本文的实验部分(见第 4 节).

### 3.2 计算启发式

SPP 在计算状态到目标的距离启发值时也采用类似的 A*算法框架.为了避免重复描述,在图 6 所显示的算法中已省略了 A*算法的实现细节,而突显启发值计算的重要步骤.除了采用上述的剪枝技术以外,在搜索到已有启发值的节点时会利用该值进行简单加和.在图 6 所示的算法 2 中,第 3 行~第 5 行做初始化和检查工作:如果在之前的搜索中已存储此启发值信息,则可以直接返回;如果当前状态包含目标,则距离为 0.第 8 行进行剪枝:如果搜索深度超出设定的阈值,为保障搜索过程继续,则以一个大值返回.第 9 行调整当前搜索深度.第 11 行、第 12 行在找到目标后进行回溯求解.第 13 行~第 23 行扩展当前节点,处理邻居状态.其中,第 16 行利用已有信息:如果邻居状态节点到目标的距离估值已经存在,则可立刻返回原状态到目标的距离估值(即当前节点的邻居状态的启发值加上搜索深度);否则,调整邻居状态的启发值.在具体实现时,如加上领域名和问题名变量作为输入参数,则可使得算法变为通用型的.

### 3.3 邻居发现算法

由于是在数据库中实现这些算法,在具体子程序中使用了大量的 SQL(standard query language)语句和技巧.这里仅以邻居发现算法为例,说明 SPP 是如何运用这些技巧的.整个邻居发现算法如图 7 所示.

算法 3 首先判断状态 $s$ 是否已经扩展过,即第 3 行、第 4 行.然后,对于未扩展的新状态会枚举所有可应用的基本动作,即第 5 行、第 6 行,其中,第 6 行使用了 SQL 特有的内部连接(INNER JOIN)操作.接下来,在第 7 行、第 8 行进行状态的演进,在第 8 行利用了 SQL 声明性的特点,使得仅用 1 句包含 UNION 以及 MINUS 等算子的 SQL 语句就实现了复杂的集合操作.第 9 行、第 10 行处理了可能的冗余问题,即外存算法中常见的重复节点检测问题.值得一提的是,像经典规划器一样,SPP 给每个命题一个魔数(magic number).由命题组成的状态也具有魔数,其值等于所包含的命题的魔数之和.第 9 行的 SQL 语句中正是利用了这个魔数和 SUM 函数实现了状态的快速识别.最后,在第 11 行、第 12 行找出了所有邻居节点,并将它们返回.

### 3.4 实 现

开发和实验环境是在 1.3GHz Celeron CPU+383 MB 内存的 PC 工作站上进行的.操作系统是 FreeBSD 6.2,数据库及版本是 Mysql5.1.22.

为了及时跟进 PDDL 的发展和利用规划研究的新进展,本文采用开源的工具 VAL 进行 PDDL 解析.VAL[24]是开源的规划验证工具,支持 PDDL3 和一些附加特性.本文中,目前仅用到其解析 STRIPS 部分.在数据库的构建方面,对全局实体要在解析之前生成,对于领域实体要在解析领域文件之后解析问题文件之前生成,最后生成问题实体.其中,生成领域实体后需要对类型之间的关系进行编译,生成问题实体后需要对命题和基本动作实例化.除了上述语法解析以外,SPP 完全基于存储过程实现,其主要算法即算法 1~算法 3.以上步骤可理解为:先基于 VAL 进行 PDDL 语法解析,然后构建数据库所需的各种实体.与现有的外存规划不同,SPP 理论上是一个完全基于外存的规划器.从读取规划定义文件开始,整个存储就是基于外存数据库的.对 VAL 进行修改以将规划领域描述入库会涉及到大量的编程工作,具体细节这里不再赘述.

在算法 1 中,初始的最大搜索深度,即 $maxH$,是一个输入值.在实际开发中,我们采用一个较小的估计值(比如 10)作为初始值,如果搜索失败,就增加这个估计值重新搜索,即迭代式的增量搜索.

实际上,这种方法是 SATPlan[10]中方法的变形.不过,一个很大的区别在于,SATPlan 中失败的搜索过程除了

知道规划解长度超过某个界限值之外没有其他价值,而 SPP 即使是搜索失败,也可以保存搜索过程中的状态演进和启发值以便以后检索,从而实现重用搜索信息的目的.

数据库的性能可以从多方面得到优化,但是本文仅采用了索引技术.数据库性能优化的方法包括索引、条块化、换页、缓存、集群、出发、列重排序、函数索引等等.数据库的 I/O 仍然是瓶颈之一,因此,采用条块化、缓存等技术是最重要的优化方式.当然,这与具体数据库的实现关系很大.但是考虑到本文是从方法上去验证采用数据库进行直接规划求解的初步工作,并且本文的重点是验证方法而不在于进行不同数据库的不同实现方式的比较,因此,本文仅采用了最基本、最常见的索引方式来进行性能优化.在未来的工作中,SPP 计划采用更多的数据库优化技术.

## 4 实验与分析

本文的实验工作包括以下两个方面:

(1) 对 IPC 基准问题进行测试,以验证相对于传统最优规划系统,SPP 能够求解更大规模的问题;

(2) 对可重用信息的问题进行测试,以验证 SPP 可以利用已有信息加快求解速度.

### 4.1 基准问题测试

考察 SPP 是否相对于传统最优规划器提高了可扩展性的问题,可以采用昂贵的机器和超大规模的问题,也可以找到一个配置较低的机器,使得在其上传统最优规划器无法求解而 SPP 可以求解.前一种方法是比较常见的,但是它会带来一个问题,即可能由于编程技巧而造成巨大的差距以至于掩盖了事物的本原.特别是在 2006 年以后多核 CPU 的出现,在 IPC-5 中就明确提出禁止使用多进程等手段.但可能是由于难以检测,到 IPC-6 时开始强制开源.为了避免上述问题,本文采用后一种办法,即低配置的机器.

在 IPC-3 中,Zeno travel 领域模拟了航空运输,其问题实例包括在多个城市之间运送旅客.对该领域的最开始 5 个问题(pfile1~pfile5,http://planning.cis.strath.ac.uk/competition/),SATPlan[10]及其后继版本 SATPlan2006 在本文的测试机器上无法求解,报告“Memory limit exceeded”错误,即有代表性的传统最优规划器无法在低配置机器上进行求解.

SPP 对这 5 个问题的求解情况见表 1.该表列出了各个问题的求解时间(反映求解速度)和扩展节点数(反映状态空间大小).从表 1 可见,SPP 能够求解这 5 个问题,并且由于剪枝技术,其真正扩展的节点数并不多.

**Table 1** Results for SPP to solve some Zeno travel planning problems

**表 1** SPP 对 IPC-3 中 Zeno travel 领域部分问题求解情况

| Problem ID | pfile1 | pfile2 | pfile3 | pfile4 | pfile5 |
|---|---|---|---|---|---|
| Time | 1s | 13s | 37min. | 14min. | 85min. |
| Node number | 2 | 8 | 96 | 53 | 259 |

然而从实验中发现,SPP 的速度受到实验机器的配置特别是内存容量的影响,即过小的内存导致了频繁的内外存交换,使得求解这些基准问题时 SPP 的速度显得比较慢.从原理上来说,这是时间换空间的必然结果,不过我们仍然会将速度的提升作为本文的未来工作之一.

此外,图 8 还给出了 SPP 基于 ER 模型(见第 2 节)所生成的 Zeno travel 领域及其问题 pfile1 的实际数据库表.左图显示了部分全局实体表、领域实体表和问题实体表,其中,D1 表示 Zeno travel 领域,P1 表示问题 pfile1.图 8 中右图给出了领域实体表 D1_parent_type 的具体内容.该表表示了领域中类型的继承关系.它实际上对应图 8 中右下图的类型关系树.

如前所述(见第 2 节关于建模正确性的讨论),基于 ER 模型将领域和问题转化为实际数据库表并由规划系统正确求解,是以实例的方式验证了建模的正确性.

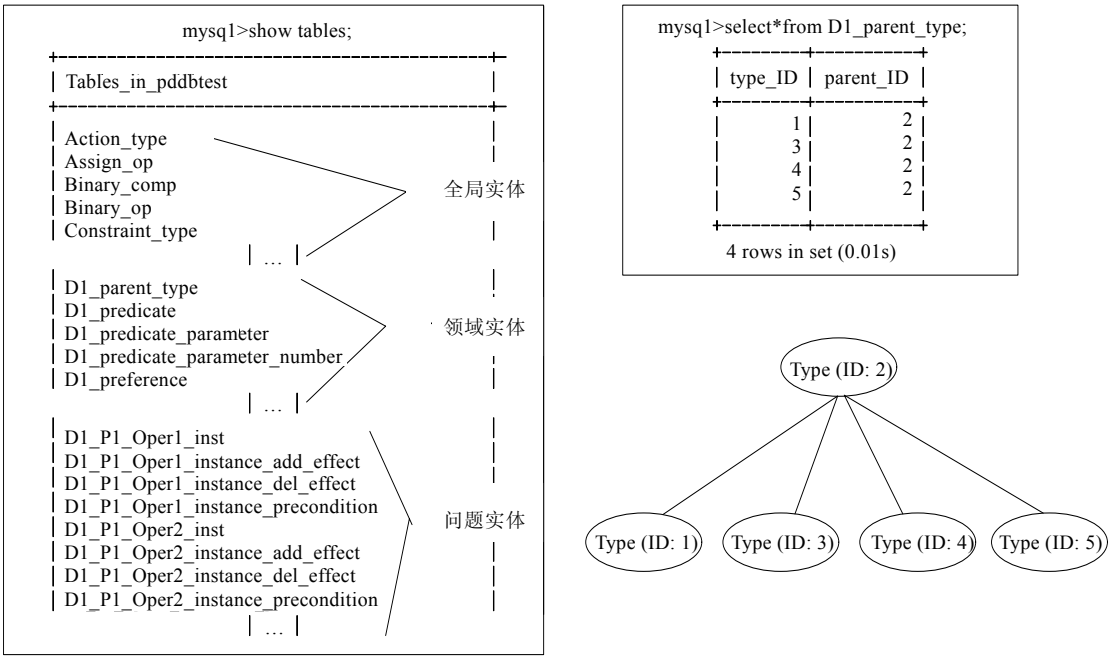


Fig.8 Practical database tables produced by SPP for the domain Zeno travel and its problem pfile1

图 8 由 SPP 产生的 Zeno travel 领域及其问题 pfile1 的实际数据库表

### 4.2 重用信息测试

为了验证 SPP 可以利用已有信息加快求解速度,本文对 Zeno travle 领域的一个具体问题 pfile3 进行了处理,以得到一组具有相似信息的问题实例.具体来说,就是构造一组具有相同的领域属性和对象数量的规划问题,考察在求解其中一个或多个问题后 SPP 的求解速度提升情况.

例 2:IPC-3 中,Zeno travel 领域的 pfile3 问题描述如下:

```
(define (problem ZTRAVEL-2-4)
(:domain zeno-travel)
(:objects
    plane1-aircraft
    plane2-aircraft
    person1-person
    person2-person
    person3-person
    person4-person
    city0-city
    city1-city
    city2-city
    fl0-flevel
    fl1-flevel
    fl2-flevel
    fl3-flevel
```

```
        fl4-flevel
        fl5-flevel
        fl6-flevel
        )
    (:init
        (at plane1 city0)
        (fuel-level plane1 fl4)
        (at plane2 city2)
        (fuel-level plane2 fl5)
        (at person1 city0)
        (at person2 city0)
        (at person3 city1)
        (at person4 city1)
        (next fl0 fl1)
        (next fl1 fl2)
        (next fl2 fl3)
        (next fl3 fl4)
        (next fl4 fl5)
        (next fl5 fl6)
    )
    (:goal (and
        (at plane2 city2)
        (at person1 city1)
        (at person2 city0)
        (at person3 city0)
        (at person4 city1)
        ))
    )
```

该问题的对象包括两架飞机、4 个旅客、3 个城市.旅客需要乘坐飞机从一个城市到达另一个城市.保持这些对象的种类和数目不变,即问题常量一致,依次移除原问题中的一个原子目标,可构造出下述 5 个原规划问题的子问题:

① (:goal (and (at person1 city1)
(at person2 city0)
(at person3 city0)
(at person4 city1)))

② (:goal (and (at plane2 city2)
(at person2 city0)
(at person3 city0)
(at person4 city1)))

③ (:goal (and (at plane2 city2)
(at person1 city1)
(at person3 city0)

(at person4 city1)))

④ (:goal (and (at plane2 city2)
(at person1 city1)
(at person2 city0)
(at person4 city1)))

⑤ (:goal (and (at plane2 city2)
(at person1 city1)
(at person2 city0)
(at person3 city0)))

子问题和原问题的目标状态不同,自然所要求的规划解也不同.调用 SPP,依次求解上述原问题和 5 个子问题,求解结果见表 2.

**Table 2** Results for SPP to solve problem instances in Example 2 by reusing information

**表 2** SPP 对例 2 构造的可重用信息问题求解情况

| Problem | Time | Expended nodes No. |
|---|---|---|
| pfile3 | 37min. | 96 |
| ① | 36s | 32 |
| ② | 4s | 5 |
| ③ | 3min. | 96 |
| ④ | 3s | 4 |
| ⑤ | 2min. | 96 |

可以看出,SPP 在很大程度上重用了求解过程的状态信息和搜索信息.求解原问题花了 37min.,而求解每个子问题的时间不超过 3min.,最少时间是 3s.类似地,对同一组数据反复实验(如对表 1 中的 5 个问题求解多次,由于信息重用,每次求解的时间会越来越短).此外还可以看到,尽可能地重用信息并非意味着一定能够减少问题的求解时间,在一定程度上还依赖于问题本身的难易程度.比如,子问题②就比子问题③容易,因为在子问题②的目标状态所涉及的 3 名旅客中,有两名旅客(person2 和 person4)的出发地和目的地是相同的,不需要搬运,而子问题③中只有 1 名旅客(person4)不需要搬运.但显然,SPP 的求解速度仍然不够快,这与其状态空间大并且全部写硬盘有关.

以上实验环节说明,在低配置的机器环境下,SPP 通过数据库的存储空间可以求解其他规划系统(如 SATPlan)因受内存限制而不能求解的规划问题.另一方面,通过搜索信息重用,SPP 可以加速求解领域常量相同的不同规划问题.

## 5 结 语

向应用拓展是智能规划研究领域的重点,而规模问题又是其中的重点之一.规划在近 20 年得到了多方面的发展,包括向不确定规划[25]拓展、寻找应用点等,但最终其重点肯定是向应用靠拢,而应用中的一大问题就是目前的基准问题普遍规模不够大.与以往的规划器不同,本文是最早使用数据库进行完全意义上的规划求解的.为了最终解决处理大规模问题的空间问题,在对规划领域建立 ER 模型之后,本文不但采用了数据库进行存储规划过程的信息,而且采用了完全在数据库内运行的存储过程作为基础语言.有了这样的基础设施,SPP 能够最大限度地利用数据库的各种功能,比如索引、外键形成的完整性约束等.更重要的是,它还可以随着数据库的发展而发展,随着数据库的扩展而扩展.比如云计算[26],在这种情况下,不是完全在数据库内部运行的程序就难以充分利用其复杂的负载均衡机制.

本文的主要创新点之一在于应用了用空间来换取时间的一般原理来解决规划应用中遇到的规模问题.首先,对计算机算法而言,空间换时间和时间换空间是哲学层面的真理.这就意味着无论什么人在多久之后的将来,要想解决规划中的空间问题都必定要在时间上做出牺牲.其次,在空间换时间方面做得最好的技术或者说学

术领域就是数据库.经过几十年的发展,数据库技术从建模到物理实现都有了很多里程碑式的成果,也有了很多具有压倒性优势的技术,比如关系数据库理论及相应系统.本文所说的利用空间换取时间,指的是利用数据库的存储空间来避免由于内存不足所导致的大规模规划问题不可解现象的产生.由于内存不足导致问题不可解在时间上可以理解为求解时间无限长.尽管由于频繁的内外存交换使得 SPP 的求解时间不短,但是毕竟使得这些问题在低配置机器上是可解的.利用空间换取时间,实质上是指获取了一种可求解能力.本文的实验结果也表明,在同样的机器配置下,利用数据库技术可以有效解决现有规划器由于内存空间不足而无法求解的问题.

本文的另一个主要创新点是利用了存储的信息加速求解.SPP 在 $A^*$算法中嵌入了对启发值的处理,具体来说,如果在求启发值过程中步长超过了长度阈值就进行剪枝,否则就产生新的阈值.此外,为了利用数据库中已有的信息,在求启发值过程中搜索到已有启发值的节点时,会利用该值进行简单加和.作为第 1 个在数据库中实现的规划系统,本文还提出了一些利用 SQL 语句进行求解的算法,比如邻居发现算法等.本文整体算法的时空复杂性与 A*算法的时空复杂性一致.

PDDL 的 ER 建模和基于存储过程的规划系统的提出,来自于本文作者在规划领域长期的研究积累和对发展前景的积极预见.本文迈出了彻底解决规划求解中空间问题的第一步,但还有很多进一步的工作需要做:

(1) 本文持久化地存储了规划过程信息,这些信息可以用来进行数据挖掘.实际上,这也是本文的出发点之一.在进行规划与学习的结合研究[27,28]中,我们发现一个根本的问题:数据匮乏.而要解决这一问题,就需要持久化存储规划过程信息.

(2) 目前 SPP 仅能处理 STRIPS 规划问题,而对于更多 PDDL 语言的特性还有待支持.结合本文作者已有的研究工作,可以扩展 SPP 求解不确定规划问题的能力,并且利用观测约简[25]技术来减少求解代价;可以利用触发器来处理派生谓词,使得 SPP 可以处理包含公理推理的规划问题[29].

(3) 可考虑利用本文作者已提出的命题关系图[30]来缩减启发估值的状态空间,以加快 SPP 的求解速度.

(4) SPP 需要结合大规模存储才能真正发挥威力,比如云计算平台(http://aws.amazon.com/rds/)或者大规模存储系统(http://storage.iu.edu/ mdss.shtml).

(5) SPP 目前已经比较充分地利用了 VAL 的领域分析功能[24],今后也可以结合其他领域分析技术来提取更多的领域知识.

**References**:

[1] Chiang YJ, Goodrich MT, Grove EF, Tamassia R, Vengroff DE, Vitter JS. External-Memory graph algorithms. In: Clarkson KL, ed. Proc. of the 6th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'95). New York: ACM Press, 1995. 22−24.

[2] Korf RE, Zhang WX. Divide-and-Conquer frontier search applied to optimal sequence alignment. In: Kautz HA, Porter BW, eds. Proc. of the 17th National Conf. on Artificial Intelligence (AAAI 2000). Cambridge: MIT Press, 2000. 910−916.

[3] Edelkamp S, Jabbar S, Schrödl S. External A*. In: Biundo S, Frühwirth TW, Palm G, eds. Proc. of the Advances in Artificial Intelligence, the 27th Annual German Conf. on AI (KI 2004). Berlin: Springer-Verlag, 2004. 226−240.

[4] Mehlhorn K, Meyer U. External-Memory breadth-first search with sublinear I/O. In: Möhring RH, Raman R, eds. Proc. of the 10th European Symp. on Algorithms (ESA 2002). Berlin: Springer-Verlag, 2002. 723−735. [doi: 10.1007/3-540-45749-6_63]

[5] Zhou R, Hansen E. Breadth-First heuristic search. Artificial Intelligence, 2006,170(4-5):385−408. [doi: 10.1016/j.artint.2005.12.002]

[6] Idwan S. Computing breadth first search in large graph using hmetis partitioning. European Journal of Scientific Research, 2009, 29(2):215−221.

[7] Edelkamp S, Jabbar S. Cost-Optimal external planning. In: Proc. of the 21st National Conf. on Artificial Intelligence (AAAI 2006). Menlo Park: AAAI, 2006. 821−826.

[8] Edelkamp S, Jabbar S, Bonet B. External memory value iteration. In: Boddy MS, Fox M, Thiébaux S, eds. Proc. of the 17th Int'l Conf. on Automated Planning and Scheduling (ICAPS 2007). Menlo Park: AAAI, 2007. 128−135.

[9] Dai P, Mausam M, Weld D. Partitioned external-memory value iteration. In: Fox D, Gomes CP, eds. Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008). Menlo Park: AAAI, 2008. 898−904.

[10] Kautz h, Selman B. Planning as satisfiability. In: Neumann B, ed. Proc. of the 10th European Conf. on Artificial Intelligence (ECAI'92). New York: John Wiley & Sons, 1992. 359−363.

[11] Bonet B, Geffner H. Planning as heuristic search. Artificial Intelligence, 2001,129(1):5−33. [doi: 10.1016/S0004-3702(01)00108-4]

[12] Fikes R, Nilsson N. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 1971,2(3-4):189−208. [doi: 10.1016/0004-3702(71)90010-5]

[13] Giacomo GD, Mancini T. Scaling up reasoning about actions using relational database technology. In: McGuinness DL, Ferguson G, eds. Proc. of the 19th National Conf. on Artificial Intelligence (AAAI 2004). Menlo Park: AAAI, 2004. 245−250.

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

[15] Richter S, Westphal M. The LAMA planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research, 2010,39:127−177.

[16] Haslum P, Geffner H. Admissible heuristics for optimal planning. In: Chien S, Kambhampati S, Knoblock CA, eds. Proc. of the 5th Int'l Conf. on Artificial Intelligence Planning and Scheduling (AIPS 2000). Menlo Park: AAAI, 2000. 140−149.

[17] Katz M, Domshlak C. Structural patterns heuristics via fork decomposition. In: Rintanen J, Nebel B, Beck JC, Hansen EA, eds. Proc. of the 18th Int'l Conf. on Automated Planning and Scheduling (ICAPS 2008). Menlo Park: AAAI, 2008. 182−189.

[18] Fox M, Long D. PDDL2.1: An extension to PDDL for expressing temporal planning domains. Journal of Artificial Intelligence Research, 2003,20:61−124.

[19] Wang GR, Yu G, Shi J, Shan JD, Zheng HY. Semantic capture and schema transformation from relational databases into object-oriented semantic databases. Ruan Jian Xue Bao/Journal of Software, 1996,7:400−409 (in Chinese with English abstract). http://www.jos.org.cn/ch/reader/view_abstract.aspx?file_no=1996s157&flag=1

[20] Wang B, Yang XC, Wang GR. A top-*K* keyword search for supporting semantics in relational databases. Ruan Jian Xue Bao/Journal of Software, 2008,19(9):2362−2375 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/2362.htm [doi: 10. 3724/SP.J.1001.2008. 02362]

[21] Hatzi O, Vrakas D, Bassiliades N, Anagnostopoulos D, Vlahavas I. A visual programming system for automated problem solving. Expert Systems with Applications, 2010,37(6):4611−4625. [doi: 10.1016/j.eswa.2009.12.047]

[22] Bylander T. The computational complexity of propositional STRIPS planning. Artificial Intelligence, 1994,69(1-2):165−204. [doi: 10.1016/0004-3702(94)90077-9]

[23] Nebel B. On the compilability and expressive power of propositional planning formalisms. Journal of Artificial Intelligence Research, 2000,12:271−315.

[24] Howey R, Long D, Fox M. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In: Proc. of the 16th IEEE Int'l Conf. on Tools with Artificial Intelligence (ICTAI 2004). Washington: IEEE Computer Society, 2004. 294−301. [doi: 10.110/ICTAI.2004.120]

[25] Rao DN, Jiang ZH, Jiang YF, Zhu HQ. Further research on observation reduction in non-deterministic planning. Ruan Jian Xue Bao/Journal of Software, 2009,20(5):1254−1268 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3453.htm [doi: 10.3724/SP.J.1001.2009.03453]

[26] Hayes B. Cloud computing. Communications of the ACM, 2008,51(7):9−11. [doi: 10.1145/1364782.1364786]

[27] Rao DN, Jiang ZH, Jiang YF, Liu Q. Learning first-order rules for derived predicates from plan examples. Chinese Journal of Computers, 2010,33(2):251−266 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2010.00251]

[28] Rao DN, Jiang ZH, Jiang YF, Wu KH. Learning non-deterministic action models for Web services from WSBPEL programs. Journal of Computer Research and Development, 2010,47(3):445−454 (in Chinese with English abstract).

[29] Jiang ZH, Jiang YF. An improved method for calculating activation sets of action derived preconditions. Chinese Journal of Computers, 2007,30(12):2061−2073 (in Chinese with English abstract).

[30] Jiang ZH, Rao DN, Jiang YF, Zhu HQ. Constructing goal agenda and macro actions using proposition relation graphs in planning. Ruan Jian Xue Bao/Journal of Software, 2011,22(1):44−56 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3861.htm [doi: 10.3724/SP.J.1001.2011.03861]

**附中文参考文献**:

[19] 王国仁,于戈,石晶,单吉第,郑怀远.从关系数据库到面向对象语义数据库的语义捕捉和模式转换.软件学报,1996,7:400−409. http://www.jos.org.cn/ch/reader/view_abstract.aspx?file_no=1996s157&flag=1

[20] 王斌,杨晓春,王国仁.关系数据库中支持语义的 Top-*K* 关键字搜索.软件学报,2008,19(9):2362−2375. http://www.jos.org.cn/1000-9825/19/2362.htm [doi: 10.3724/SP.J.1001.2008.02362]

[25] 饶东宁,蒋志华,姜云飞,朱慧泉.对不确定规划中观测约简的进一步研究.软件学报,2009,20(5):1254−1268. http://www.jos.org.cn/1000-9825/3453.htm [doi: 10.3724/SP.J.1001.2009.03453]

[27] 饶东宁,蒋志华,姜云飞,刘强.从规划解中学习一阶派生谓词规则.计算机学报,2010,33(2):251−266. [doi: 10.3724/SP.J.1016.2010.00251]

[28] 饶东宁,蒋志华,姜云飞,吴康恒.从 WSBPEL 程序中学习 Web 服务的不确定动作模型.计算机研究与发展,2010,47(3):445−454.

[29] 蒋志华,姜云飞.一种计算动作派生前提的激活集的改进方法.计算机学报,2007,30(12):2061−2073.

[30] 蒋志华,饶东宁,姜云飞,朱慧泉.利用规划命题关系图构建目标议程和宏动作.软件学报,2011,22(1):44−56. http://www.jos.org.cn/1000-9825/3861.htm [doi: 10.3724/SP.J.1001.2011.03861]

**饶东宁**(1977－),男,广东兴宁人,博士,副教授,CCF 会员,主要研究领域为智能规划,图论.

E-mail: raodn@gdut.edu.cn

**蒋志华**(1978－),女,博士,副教授,主要研究领域为智能规划.

E-mail: tjiangzhh@jnu.edu.cn

**姜云飞**(1945－),男,教授,博士生导师,主要研究领域为定理机器证明,智能诊断,智能规划.

E-mail: issjyf@mail.sysu.edu.cn