基于单逻辑程序的一致性规划任务有限域表示方法*

李伟生，刘森森
（重庆邮电大学 计算机科学与技术学院，重庆 400065）

摘　要：有限域表示（FDR）能有效地压缩状态空间，其转换算法在实例化阶段对每个初始状态都生成一个逻辑程序，而一致性规划任务的初始状态数量通常较大，所以这通常需要较大的时间和空间开销，甚至导致内存溢出。为了提高转换算法运行效率使其能处理更为复杂的规划问题，提出了一种基于单逻辑程序的 IFDR 转换算法。IFDR 算法从初始信念状态中所有可能的初始世界状态得到一个事实集，再由动作和公理计算得到一个规则集。一个事实集和一个规则集组成一个逻辑程序，IFDR 用此单逻辑程序完成实例化。实验结果表明 IFDR 算法在解决问题的效率和数量上都有所提高。

关键词：一致性规划；有限域表示；数据记录搜索算法；逻辑程序；PPDDL；信念状态
中图分类号：TP181　　**文献标志码**：A　　**文章编号**：1001-3695(2014)02-0404-04
doi:10.3969/j.issn.1001-3695.2014.02.020

Finite-domain representations of conformant planning tasks based on single logic program

LI Wei-sheng, LIU Sen-sen
(*College of Computer Science & Technology*, *Chongqing University of Posts & Telecommunications*, *Chongqing* 400065, *China*)

Abstract: Finite-domain representations (FDR) is one of the best methods to compress the size of state space, but it requires building a logic program for each initial state in grounding stage. Unfortunately the number of the possible initial state in conformant planning is always huge. So there will be heavy time and space overhead and even leads to memory overflow. In order to improve the efficiency of the algorithm, this paper designed a new method named IFDR using a single logic program to translate the PPDDL task into grounded one. A logic program consists of a set of facts and a set of rules. The set of facts of the logic program was formed by the atoms in all the possible initial state, and the set of rules was derived from the axiom and operator definitions. The outcomes of comparative experiments validate the accuracy and effectiveness of the new algorithm.
Key words: conformant planning; FDR; datalog exploration; logic program; PPDDL; believe state

0　引言

现实中多数规划问题并不满足经典规划的条件，都存在不确定性，如初始状态不完全可知、动作的执行效果不能完全预料，环境也会发生不可预知的变化等。在不确定的情况下寻求规划解称之为强规划。一致性规划（conformant planning）是强规划的一种，又称为完全不可观察强规划，即所有状态变量都不是观察变量，在运行时不能得到任何信息[1]。

一致性规划系统的性能受多个因素的影响，其中一个主要因素就是如何处理初始状态的不确定性[2]，即如何表示初始信念状态。初始信念状态即所有可能的初始世界状态的集合。对信念状态的表示从两个方面影响一致性规划系统的性能[3]：a）如果一个信念状态中世界状态数量过多，将增大内存消耗，最终可能导致内存溢出；b）初始信念状态的个数直接影响计算后继信念状态的时间复杂度。

目前已有多种信念状态表示方法。CFF[4] 规划系统采用的方法是保存到达某个信念状态的动作序列，而不是明确表示出此信念状态。当需要时再用初始状态和动作序列计算出此信念状态。POND[5] 规划系统则使用有序二元决策图（ordered binary decision diagram，OBDD）来表示初始信念状态，并用有序二元决策图库（OBDD-library）计算后继信念状态。T0[6] 则是将一致性规划任务转换成经典规划任务，再用经典规划系统 FF[7] 求解，最终再将规划解转换成一致性规划解的形式。

CPA[8] 与以往的规划系统不同，它在搜索过程中仅保存一个近似的状态集合。一个信念状态用它所包含状态的交集近似表示。这种方法的优点是后继信念状态可以在线性时间内计算出来。但代价是要频繁计算初始近似状态，造成开销增大。与 CPA 类似，DNF 和 CNF 两个规划系统中也使用近似表示方法，DNF[2] 用析取范式表示信念状态，而 CNF[4] 用合取范式表示信念状态。

在文献[9]中提出一种将 PPDDL[10] 表示的一致性规划任务转换成有限域表示形式的 CPT-FDR 方法，实现了经典规划的有限域表示（finite-domain representations，FDR）[11] 方法在一致性规划中的扩展。FDR 是基于 SAS^+ 表示形式的扩展，它使用有限域变量来表示世界状态，相比二元变量能够有效地压缩状态空间。为进一步解决 CPT-FDR 算法在解决问题的效率和

收稿日期：2013-05-23；**修回日期**：2013-07-01　　**基金项目**：国家自然科学基金资助项目（61142011）；国家教育部新世纪优秀人才计划资助项目（NCET-11-1085）

作者简介：李伟生（1975-），男，四川安岳人，教授，博士，主要研究方向为智能信息处理、模式识别、信息融合；刘森森（1988-），男，河南焦作人，硕士研究生，主要研究方向为智能规划（liuss8723@163.com）.

数量上的不足,本文进一步改进了其中的数据记录搜索(datalog exploration)算法,提出了一种新的转换算法——IFDR 算法。该算法能有效地减少实例化阶段消耗的时间,并能提高转换规划问题的成功率。

1　CPT-FDR 转换算法

CPT-FDR 转换算法分四步:规范化、合成不变量、实例化以及生成 CPT-FDR 任务。合成不变量和实例化之间没有先后顺序。合成不变量阶段并不是必需的,但如果去掉此过程,CPT-FDR 的状态变量就与 PPDDL 的实例原子一一对应,也就变成了二元变量。

1.1　简述

规范化阶段主要完成三项任务:去除类型化、简化条件和效果。经过此阶段后得到规范的 PPDDL 任务,满足如下条件:动作的前提条件、效果条件、公理体以及目标都是文字的合取;动作效果是全称量化的简单效果的合取。

合成不变量:不变量是指所有可达的世界状态都满足的一个属性。CPT-FDR 算法中需要计算的是互斥不变量,同一互斥组内的命题不能同时为真。一个长度为 n 的互斥组可以编码为一个有 $n+1$ 取值的状态变量。其中 n 个值分别表示某个命题为真,第 $n+1$ 个值表示 n 个命题全部为假。

实例化简单来讲就是用对象取代目标、公理和动作中的变量。例如,设一个规划任务有 n 个对象,某个动作有 k 个参数变量,则每个变量有 n 种取值。该动作实例化后有 n^k 个实例。但这种方法太过原始,得到的实例中大部分不可用。后面将详细叙述改进的算法。

生成任务阶段:CPT-FDR 任务用五元组表示$\Pi = \langle V, s_0, s_*, A, O\rangle$,利用前面得到的信息先生成 CPT-FDR 变量,再转换成其他要素,生成 CFRD 任务。

1.2　数据记录搜索算法

改进的实例化算法的基本思想是先由数据记录搜索算法求得放松规划任务[11] $R(\Pi)$的可达原子集β,再用β实例化公理和动作。数据记录搜索算法分为三个步骤:生成逻辑程序、将逻辑程序转成标准形式(为计算范型作准备)、计算范型。

定义 1　逻辑程序(logic program),也称数据记录程序。

逻辑程序是一个二元组$\langle F, R\rangle$。其中 F 是实例化原子集合,也称事实集;R 是数据记录规则集合,简称规则集。数据记录规则是一个一阶公式,形如 $\varphi_1 \wedge \cdots \wedge \varphi_k \rightarrow \psi (k \geqslant 0)$,其中 $\varphi_1, \cdots, \varphi_k$ 为规则主体,ψ 为规则头部。

生成逻辑程序即求事实集 F 和规则集 R。事实集 F 由初始状态中的原子组成。规则集 R 是由公理、动作和目标公式推导出来。规范化后 PDDL 任务中条件都是合取形式,现用 α^+ 表示条件中的正文字。计算 R 的具体方法如下:

对公理 $a = (\alpha \leftarrow \psi)$ 其中 $\psi^+ = \psi_1^+ \wedge \psi_2^+ \wedge \cdots \wedge \psi_m^+$ 且 free$(\alpha) \vee$ free$(\psi) = \{X_1 \cdots X_k\}$,引入公理可用性规则 a-applicable$(X_1 \cdots X_k) \leftarrow \psi^+$ 和效果规则 $\alpha \leftarrow a$-applicable$(X_1 \cdots X_k)$。

对动作 o,设其参数列表为 $\{X_1 \cdots X_k\}$,前提条件为 α 且 $\alpha^+ = \alpha_1^+ \wedge \alpha_2^+ \wedge \cdots \wedge \alpha_m^+$,引入动作可用性规则 o-applicable$(X_1 \cdots X_k) \leftarrow \alpha^+$。对于此动作的添加效果 e,设其添加的原子为 ψ,ψ 的取值范围为 $\{Y_1 \cdots Y_l\}$,效果条件为 χ,且 $\chi^+ = \chi_1^+ \wedge \chi_2^+ \wedge \cdots \wedge \chi_n^+$,引入效果触发规则 e-triggered$(X_1 \cdots X_k, Y_1 \cdots Y_l) \leftarrow o$-applicable$(X_1 \cdots X_k), \chi^+$,另外还要引入效果规则 $\psi \leftarrow e$-triggered$(X_1 \cdots X_k, Y_1 \cdots Y_l)$。

对目标 α,其中 $\alpha^+ = \alpha_1^+ \wedge \alpha_2^+ \wedge \cdots \wedge \alpha_m^+$,引入目标规则 goal-reachable$() \leftarrow \alpha^+$。

1.3　计算逻辑程序的标准形式

所谓标准形式即逻辑程序中所有规则要么是投影规则要么是连接规则。

投影规则是一元规则,形如 $\alpha \leftarrow \alpha_1$,规则头部的变量集合是规则体变量集合的子集。

连接规则是二元规则,形如 $\alpha \leftarrow \alpha_1, \alpha_2$,规则头部的变量一定也在规则体中,在规则体中而不在头部中的变量一定同时出现在 $\alpha_1 \alpha_2$ 中。

将逻辑程序转换成标准形式的规则:

若规则中的原子含有重复变量,如 atom(X, X),将其中一个变量重新命名,得 atom(X, X')。再向规则体中添加新原子 equal(X, X'),直至没有重复变量。另外对于规划任务中的每一个对象 o,生成事实 equal(o, o),添加到事实集。

若变量 X 只出现在规则头部(未出现在规则体中),则向规则体中添加原子 object(X)。

若规则体为空,则将规则转换为事实原子。经过前面两步转换,出现在头部中的变量一定出现在主体中,现在主体为空,显然头部中的原子应不含变量即为实例化原子。

经过上述转换后,所有一元规则都已是投影规则。接下来要将二元变量转换成连接规则,再将多元连接规则转换为二元的。

计算标准形式:在前面两步工作的基础上,采用增量方法计算可达事实,设置变量跟踪可达事实,利用索引结构进行规则匹配[9]。数据记录程序的范型是一个可达原子集合。

2　CPT-FDR 算法的改进

2.1　CPT-FDR 存在的不足

CPT-FDR 算法是将 PPDDL 表示的一致性规划任务转换成有限域表示。它实现了 FDR 在一致性规划中的扩展,能够很好地压缩一致性规划任务的状态空间。但 CPT-FDR 还存在一些不足之处,主要表现为算法运行效率低、耗时较长、解决的问题个数少。以 IPC-5 的 sortnet 规划域为例,该规划域有 15 个问题,CPT-FDR 算法转换第 9 个问题文件时已超过 600 s,而第 10 问题则因出现内存错误而终止程序。

2.2　对 CPT-FDR 的改进

程序运行结果显示,实例化阶段的运行时间占总时间的很大比例(见后文的实验数据)。实例化阶段的核心是数据记录搜索算法,本文提出的 IFDR 算法对 CPT-FDR 中数据记录搜索算法进行了改进,能够较大程度地减少实例化阶段消耗的时间,并提高解决规划问题的成功率。

在一致性规划中可能存在多个初始世界状态,CPT-FDR 算法中从每一个初始世界状态计算一个事实集合,由公理、动作和目标公式计算出一个规则集合。也就是说有 n 个初始世界状态就要生成 n 个逻辑程序,它们的规则集是相同的。

这样做的缺点是当 n 值较大时数据记录搜索算法的时间和空间开销就很大,而一致性规划任务的初始状态的数量往往很庞大。以 IPC-6 的 dispose 规划域为例,此规划域描述的任

务是回收垃圾。垃圾的数量以及地点个数已知,垃圾箱的位置确定,但垃圾的位置不确定。dispose-*m*-*n* 规划问题表示有 $m \times m$ 个点,*n* 个垃圾对象,$m \times m$ 个顶点组成网格连通图。dispose-*m*-*n* 规划问题所有可能的初始状态个数为$(m \times m)^n$。显然随着 *m* 和 *n* 的增大,其初始状态个数将急剧增多。

数据记录算法的最终目的是为了求放松规划任务的可达原子集,本文提出的改进方法是将所有的初始世界状态中的原子集求并集,得到一个事实集。由于 CPT-FDR 生成的 *n* 个逻辑程序规则集都相同,本文取其规则集。这样一个规则集一个事实集,只需生成一个逻辑程序。下面给出生成逻辑程序算法伪代码:

```
Input: normalized PPDDL task
Output: logic program
compute the set of facts:F
factSet = ∅
for init in task.init:
    factSet = factSet ∪ set(init)
compute the set of rules: R
ruleSet = compute ruleSet following the rules described
above
prolog = construct a object of PrologProgram class
using factSet and ruleSet
return prolog
```

计算范型时 CPT-FDR 算法需要遍历每一个逻辑程序。现在只有一个逻辑程序,本文将此逻辑程序的事实集中的原子压入队列,利用索引结构依次对每一个原子进行规则匹配。计算范型算法伪代码如下:

```
Input: logic program
Output: the canonical model of logic program
for each join rule r in ruleSet:
    r.index1 = construct an empty hashtable
    r.index2 = construct an empty hashtable
unifier = Unifier(ruleSet)
queue = construct an empty queue
result = ∅
Enqueue all facts in factSet
Enqueue a fact means that add it to queue and result if
it is not yet a element of result
while queue is not empty:
    next_atom = queue.pop()
    matches = unifier.unify(next_atom)
    for match in matches:
        if match refers to α1 in r = α←α1:
            let φ be the variable assignment
            and φ(α1) = next_atom
            Enqueue φ(α)
        else if match refers to α1 in r = α←α1,α2:
            let φ be the variable assignment
            and φ(α1) = next_atom
            key = φ and φ∈free(α1)∩free(α2)
            add φ to r.index1[key]
            Enqueue (φ∪γ)(α) for each
            γ∈r.index2[key]
        else if match refers to α2 in r = α←α1,α2:
            similar to the second case
```

result 用来存放当前可达的原子,算法执行的过程中不断将确定可达的原子加入其中。queue 是用来追踪已被确定可达但还未对规则体中的条件进行匹配的原子。本文称 result 中的原子是闭合的,而 queue 中的原子是开放的。

unifier 是一个索引结构,用于查询 prolog。对于一个实例化的原子 *a*,unifier 能够确定所有满足 α_1 或 α_2 与 *a* 统一的投影规则 $\alpha \leftarrow \alpha_1$ 和连接规则 $\alpha \leftarrow \alpha_1, \alpha_2$,换而言之,可以使用与获取 *a* 相同的方法来实例化 α_1、α_2。

每一个连接规则 $\alpha \leftarrow \alpha_1, \alpha_2$ 都有两个哈希表 $index_1$、$index_2$。它们分别是 $\alpha_1\alpha_2$ 交集中原子的实例化形式到 α_1 和 α_2 中变量的映射。这两个哈希表索引能够很方便地找出 α_2 所有可能和 α_1 实例相匹配的实例,反之亦然。

3 实验结果及分析

本文用 Python 脚本语言实现了对 CPT-FDR 算法的改进,并在 Linux 平台下作了对比实验。本文测试的规划域主要来自国际智能规划大赛第五届(IPC-5)和第六届(IPC-6)的标准规划域以及 T0 规划系统的测试域。其中 comm、sortnet、uts(5)来自 IPC-5,在数据表中分别用 cm、sort、u(5) 表示;forest、uts(6)、blocks 来自 IPC-6,在数据表中分别用 f、u(6)、block 表示;butc、safe 以及 grid 则是 T0 规划系统的测试域,分别用 bu、safe、grid 表示。另外还与 LAMA[12] 规划系统的 translate 部分进行了实验对比,LAMA 是经典规划中一个具有代表性的规划系统。具体的实验环境如下,处理器 AMD Athlon™ Dual Core Processor 5000B,内存 2 GB,操作系统 Ubuntu10.10。

3.1 CPT-FDR 改进前后实验对比

CPT-FDR 算法分为四部分,本文主要考察 CPT-FDR 消耗的总时间和实例化阶段的时间,单位为 s。限定运行时间为 600 s,超过 600 s 则终止程序。限于篇幅,在此不对各规划域每个问题的运行结果一一列举。表格中"—"表示时间明显超过 600 s。总时间对比如表 1 所示。

表 1 总时间对比 /s

规划问题	CPT-FDR	IFDR	规划问题	CPT-FDR	IFDR
safe-50	1.797	0.125	cm-05	2.625	0.141
safe-70	4.500	0.219	cm-10	249.547	5.547
safe-100	12.391	0.406	cm-11	601.656	11.937
block-03	0.406	0.062	cm-15	—	250.672
block-04	4.250	0.547	u(6)-05	3.328	0.281
block-05	59.109	14.875	u(6)-09	227.188	12.672
c-60-30	1.547	0.562	u(6)-10	—	31.891
c-91-45	3.125	1.172	u(6)-13	—	445.500
c-119-59	4.828	1.969	bu-50	12.312	0.266
u(5)-l10	4.562	0.156	bu-100	88.312	1.078
u(5)-r10	12.250	0.453	bu-150	287.781	2.297
u(5)-k10	28.172	0.828	bu-300	—	8.906

从表 1 可以看出,CPT-FDR 转换算法在 uts(5)、safe、blocks 以及 corner-cube 等规划域上表现良好,其运行时间大部分在 10 s 以内,但在 comm 等规划域上其运行速度并不理想。当然在规划问题比较简单时,CPT-FDR 的时间消耗不是很大,如 comm-01 至 comm-05 的运行时间都少于 10 s。然而随着规划问题越来越复杂,其花费的时间也迅速增加。仍以 comm 域

为例,规划问题 comm-11 的运行时间在 600 s 左右,而 comm-13 耗时则超过半个小时,这显然是不合理的。改进后的 IFDR 算法在这些规划域上的运行效率有了显著提高。当规划问题较为复杂时对比效果更为明显,这是因为一般来讲,问题越复杂初始状态个数也越多,CPT-FDR 的实例化算法的缺点也就越明显。

表 2 中给出了部分规划域改进前后实例化阶段的时间消耗对比。对比数据表 1 可得出如下结论:原始的 CPT-FDR 算法中实例化阶段消耗的时间占总时间的 60% ~90% 左右。而改进后的算法,实例化阶段耗时有了明显下降。

表 2　实例化阶段时间对比　　/s

规划问题	CPT-FDR	IFDR	规划问题	CPT-FDR	IFDR
cm-05	2.422	0.015	f-05	320.031	0.375
cm-08	39.485	0.047	f-06	460.625	0.469
cm-10	233.469	0.141	f-08	—	0.609
cm-11	559.672	2.263	f-09	—	0.703
cm-15	—	4.699	sort-05	1.203	0.031
u(6)-05	2.876	0.031	sort-08	195.281	0.187
u(6)-09	183.703	0.593	sort-09	—	3.438
u(6)-10	—	1.374	sort-15	—	142.797
u(6)-13	—	18.360	bu-50	12.312	0.094
grid-2-5	8.031	0.203	bu-100	85.203	0.328
grid-3-5	4.266	0.156	bu-150	278.813	0.687
grid-4-5	88.419	1.109	bu-300	—	8.984

表 3 给出了 CPT-FDR 算法改进前后所能解决的问题个数对比情况,其中规划域名"/"之后的数字为该规划域中的问题个数。这里不能解决的问题是指在给予充足时间的情况下,出现内存错误。可以看出改进之后 CPT-FDR 能够解决更多的规划问题,当然有些规划域的规划问题仍不能全部解决。这是因为明确表示出每一个初始状态是规范化阶段任务,更多规划问题在规范化阶段因出现内存错误而终止程序。

表 3　改进前后解决问题个数对比　　/s

规划域	CPT-FDR	IFDR	规划域	CPT-FDR	IFDR
u(5)/30	30	30	block/5	4	4
cm/25	11	15	u(6)/28	8	13
sort/15	8	15	f/8	8	8

3.2　与 LAMA 的实例化算法对比

LAMA 规划系统基于 Fast-Downward 规划系统,同样使用有限域表示(FDR)方法来表示信念状态。接下来将 IFDR 算法和 LAMA 进行实例化阶段耗时对比,实验结果如表 4 所示。对于两个 logistic 规划域,本文对规划问题作了简单修改,以便让各个对象个数保持一致。表 4 中 log 即 logistic。

表 4　改进后 CPT-FDR 与 LAMA 中实例化耗时对比　　/s

规划问题	LAMA	IFDR	规划问题	LAMA	IFDR
block-3	0.016	0.016	log-1	0.016	0.016
block-4	0.016	0.031	log-2	0.016	0.031
block-5	0.016	0.109	log-3	0.016	0.031
block-6	0.031	—	log-4	0.062	0.047

由实验结果可知 LAMA 中实例化阶段花费时间较少。IFDR 算法在简单的规划问题上表现不错,与 LAMA 的表现相当,但当问题较为复杂时其性能要比 LAMA 差很多。比如 block-6(有 6 个木块的积木摆放问题)LAMA 仅需要 0.031 s,而 CPT-FDR 要超过 600 s。这是因为经典规划任务的初始状态是确定的,如 block 域中每个木块的初始位置是已知的,而一致性规划任务的初始状态并不确定。由于存在不确定因素,一致性规划要比经典规划复杂得多,因此需要更多的时间。

4　结束语

CPT-FDR 算法实现了经典规划领域中的有限域表示方法在一致性规划中的扩展,但 CPT-FDR 在运行效率上还存在一定的不足。本文对 CPT-FDR 实例化阶段的数据记录搜索算法进行了改进,提出了 IFDR 转换算法。由实验结果可以看出改进后的转换算法在运行效率上有了显著提高。需要指出的是,CPT-FDR 算法要求明确表示出信念状态中的每一个世界状态,这对于某些极复杂的规划问题来讲是十分困难的。CPA、CNF 等规划系统近似地表示信念状态的思想应该是今后的发展方向。

参考文献:

[1] 周俊萍, 殷明浩, 谷文祥,等. 部分可观察强规划中约减观察变量的研究[J]. 软件学报, 2009, 20(2): 290-304.

[2] TO S T, PONTELLI E, SON T C. A conformant planner with explicit disjunctive representation of belief states[C]//Proc of the 19th International Conference on Automated Planning and Scheduling. 2009: 305-312.

[3] TO S T, SON T C, PONTELLI E. A new approach to conformant planning using CNF * [C]//Proc of the 20th International Conference on Automated Planning and Scheduling. 2010:169-176.

[4] BRAFMAN R, HOFFMANN J. Conformant planning via heuristic forward search: a new approach[J]. Artificial Intelligence, 2006, 170(6-7): 507-541.

[5] BRYCEL D, KAMBHAMPATI S, SMITH D E. Planning graph heuristics for belief space search[J]. Journal of Artificial Intelligence Research, 2006, 26: 35-99.

[6] PALACIOS H, GEFFNER H. From conformant into classical planning: efficient translations that may be efficient too[C]//Proc of the 17th International Conference on Automated Planning and Scheduling. 2007:264-271.

[7] 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.

[8] SON T C, TU P H, GELFOND M, *et al*. Conformant planning for domains with constraints:a new approach[C]//Proc of American Association for Artificial Intelligence. 2005:1211-1216.

[9] LI Wei-sheng, ZHANG Zhen, WANG Wei-xing. CPT-FDR: an approach to translating PPDDL conformant planning tasks into finite-domain representations[J]. Chinese Journal of Electronics, 2012, 21(1): 53-58.

[10] YOUNES H L S, LITTMAN M L. PPDDL1.0: the language for the probabilistic part of IPC'4[C]//Proc of International Planning Competition. 2004:167-170.

[11] HELMERT M. Concise finite-domain representations for PDDL planning tasks[J]. Artificial Intelligence, 2009, 173(5): 503-535.

[12] RICHTER S, WESTPHAL M. The LAMA planner: guiding cost-based anytime planning with landmarks[J]. Journal of Artificial Intelligence Research, 2010, 39(2): 127-177.