

基于延迟部分推理的快速前向规划系统

蔡敦波^{1),2),3)} 殷明浩^{1),2),3)} 谷文祥³⁾ 孙吉贵^{1),2)} 刘科成⁴⁾

¹⁾ (吉林大学计算机科学与技术学院 长春 130012)

²⁾ (吉林大学符号计算与知识工程教育部重点实验室 长春 130012)

³⁾ (东北师范大学计算机学院 长春 130117)

⁴⁾ (东北师范大学外国语学院 长春 130024)

摘 要 根据动作组件诱发关系的存在和抵制计算的必要性,提出一个计算松弛规划解的新方法——延迟部分推理.该方法在考虑动作删除效果的假定下,构造不包含任何互斥关系的组件规划图,通过定义“松弛诱发”关系预测后续规划过程中可能出现的组件诱发现象,在松弛规划解提取阶段判断动作组件间的“松弛诱发”关系并选择抵制动作避免可能发生的消极作用.基于延迟部分推理方法定义了新的启发式函数和剪枝策略,设计了规划系统 FFc 并在多个国际通用的测试域上进行实验.结果表明,FFc 较之 Fast-Forward 在求解效率和求解质量方面都有显著的提高.

关键词 智能规划;启发式搜索;朴素组件规划图;延迟部分推理

中图法分类号 TP301

Fast Forward Planning System Based on Delayed Partly Reasoning

CAI Dun-Bo^{1),2),3)} YIN Ming-Hao^{1),2),3)} GU Wen-Xiang³⁾ SUN Ji-Gui^{1),2)} LIU Ke-Cheng⁴⁾

¹⁾ (College of Computer Science and Technology, Jilin University, Changchun 130012)

²⁾ (Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Changchun 130012)

³⁾ (College of Computer Science, Northeast Normal University, Changchun 130117)

⁴⁾ (School of Foreign Languages, Northeast Normal University, Changchun 130024)

Abstract Heuristic based planning becomes the main trend of AI planning and has been proven to be successful in almost every type of planning problems. High quality heuristics and effective pruning methods are two keys to such planning systems. Realizing the two techniques based on relaxed-plans was first used for the Fast-Forward(FF)planning system and is still used by current top-performing planners. Concerning the inconsistent performance of FF in ADL domains, the authors introduce a new method for extracting relaxed plans while considering the inducing relations between components and the necessity of doing confrontations that are common in ADL planning. A relaxed inducing relation between components is proposed to predict possible inducing relations in the actual planning process. Based on actions' delete effects and a simplified components planning graph, confrontations are done in the relaxed-plan-extraction phase to handle negative interactions between components. Both the improved heuristic and the improved pruning technique based on the new relaxed-plan extraction method are implemented in a system called FFc. Experi-

收稿日期:2006-04-16;最终修改稿收到日期:2007-12-03. 本课题得到国家自然科学基金重大项目(60496320)、国家自然科学基金(60773097,60473003,60473042,60573067)和教育部高等学校博士学科点专项科研基金(20050183065)资助. 蔡敦波,男,1981 年生,博士研究生,主要研究方向为智能规划、约束满足问题. E-mail: dunbocai@gmail.com. 殷明浩,男,1979 年生,博士研究生,主要研究方向为智能规划、规划识别和自动推理. 谷文祥,男,1947 年生,教授,博士生导师,主要研究领域为智能规划与规划识别、模糊推理及其应用. 孙吉贵,男,1962 年生,教授,博士生导师,主要研究领域为智能规划与自动推理. 刘科成,男,1980 年生,硕士,主要研究方向为智能规划与规划识别.

mental results show FFC outperforms FF in several ADL domains in both planning efficiency and planning quality. The authors' work shows the subtleness of state space planning that handles conditional effects partially using an IPP method or factored expansion, and provides an efficient method to deal with such complications.

Keywords AI planning; heuristic search; Naïve components planning graph; delayed partly reasoning

1 引 言

在过去十多年中,智能规划研究领域取得了巨大的突破,相对于 1995 年之前的智能规划系统,现代规划系统无论在规划求解规模上还是在规划求解效率上都有数量级的提高. 为了进一步提高智能规划系统的求解效率,研究人员提出了多种智能规划求解方法,例如,基于问题转换的规划方法^[1-2]、基于启发式搜索的规划方法^[3-7]、基于因果链接的规划方法^[8-9]等. 其中,基于启发式搜索的规划方法是当前规划研究的热点. 该类规划方法通过领域无关的启发函数和剪枝策略等技术提高求解效率,在国际智能规划竞赛(International Planning Competition, IPC)^①中取得了优异的成绩,如 HSP^[3]、Fast-Forward(FF)^[4]、LPG^[5]、Fast Downward^[6]和 SGPlan^[7]等.

在多个基于启发式搜索的规划系统中,FF 是表现最为突出的智能规划系统之一. 该系统分别获得了 2000 年智能规划竞赛的“杰出性能”奖和 2002 年智能规划竞赛 STRIPS^[10] 规划的“最优性能”奖. FF 系统的核心技术包括增强爬山算法(enforced hill-climbing)、基于松弛规划解(relaxed plan)的启发式函数和“有利动作”剪枝策略(helpful actions). 此外,为了保证系统的完备,在增强爬山算法失败后该系统调用贪婪最好优先搜索算法(greedy best-first)^[11]. 由于增强爬山算法通常能够成功求解,FF 在多个规划域上显示出较高的效率^[4]. FF 的设计者 Hoffmann 从启发式函数性能的角度对该现象进行了详细的实践和理论分析^[12]. 然而,在 ADL^[13] 语言描述的规划问题上,增强爬山算法存在大量失败的情况. 实际上,FF 性能不一致的关键在于“有利动作”策略对于 ADL 语言描述的规划任务存在失效情况.

相对于 STRIPS 语言,ADL 语言具有更强的表达能力,是一个更加方便规划问题建模的语言. 但

是,ADL 语言的复杂性对规划系统的推理能力提出了更高的要求. 针对其中最为复杂的语法特性——条件效果,研究者提出了多种处理方法,如全扩展法(full expansion)^[14]、IPP 方法^[15]和分枝扩展法(factored expansion)^[16],其中,全扩展法可能导致动作数目的指数级别增加,适合在规划求解的过程中使用;后两种方法类似,均采用不完全处理的方式,适合在规划求解的过程之前使用,但是要求规划过程更加复杂^[16]. 为了在规划系统的预处理阶段降低规划任务的规模以及制定目标议程(goal agenda)^[17],FF 在 ADL 规划中采用 IPP 方法存储动作,使用图规划^[18]算法计算松弛规划任务(忽略动作删除效果)的规划解. 事实上,经过 IPP 方法和分枝扩展法处理的动作效果之间都存在“组件诱发”关系(one component induces other components)^[16]. 松弛图规划算法无法处理这种关系,导致“有利动作”策略产生错误的剪枝行为,使得 FF 系统的整体性能下降.

针对这个难题,本文定义了组件之间的“松弛诱发”关系并提出一个称为延迟部分推理的方法. 我们首先构造一个不包含任何互斥关系的组件规划图——朴素组件规划图,在之后的松弛规划解提取过程中判断“松弛诱发”关系并抵制(confront)该关系可能产生的消极作用. 在松弛规划图的第 0 时间步处理“松弛诱发”关系时,将抵制动作标记为“有利动作”,从而改进了 FF 系统的“有利动作”策略. 在此方法中,“松弛诱发”关系能够预测一部分可能出现的组件诱发关系;与已有方法在规划图扩展阶段处理可能诱发关系不同,我们将“松弛诱发”关系的处理推迟到规划解提取阶段. 因此,本文提出的延迟部分推理方法具有 3 个特点:(1) 通过处理一部分可能的组件诱发关系减少了“有利动作”策略的错误剪枝行为;(2) 计算代价低;(3) 提高了启发式函数的信息量. 我们以 FF 系统为基础,基于延迟部分推

① International Planning Competition. <http://ipc.caps-conference.org>

理方法,实现了称为 FFc 的规划系统,在 ADL 语言描述的多个标准测试域进行实验.结果表明,在大多数问题上 FFc 的求解效率和求解质量比 FF 都有显著的提高.

本文第 2 节分析 FF 在 ADL 规划上效率低下的原因;第 3 节提出“松弛诱发”关系和基于朴素组件规划图的延迟部分推理方法;第 4 节在国际上通用的 ADL 语言描述的规划域上对 FFc 和 FF 进行性能比较并对实验结果进行初步的分析;最后介绍相关研究并对全文进行总结.

2 FF 系统的局限性

FF 规划系统的结构如图 1 所示.

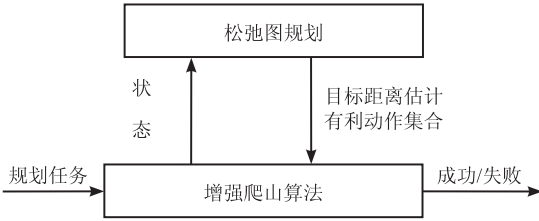


图 1 FF 的基本系统结构

松弛图规划模块是 FF 系统的核心模块之一,它一方面为系统提供当前状态与目标状态的距离估计,一方面为增强爬山算法提供有利动作以剪裁搜索空间.一个规划任务(planning task)对应的松弛规划任务(relaxed planning task)是忽略该任务中动作删除效果的结果. FF 利用图规划计算松弛规划任务的解,随后,使用该解的代价估计原规划任务的代价. 本文将求解松弛规划任务的图规划算法称为松弛图规划(relaxed Graphplan, rGP). FF 使用目标议程方法降低问题求解难度,使用松弛图规划模块提供的启发信息引导搜索,并设计了高效的局部搜索算法——增强爬山实现快速求解,仅在增强爬山算法失败时调用贪婪最好优先算法. 上述各种技术的综合使得 FF 在大部分逻辑规划问题求解中表现出优异的性能. 然而,我们发现 FF 系统在求解 ADL 规划问题时,与求解 STRIPS 规划问题相比,虽然在较小规模问题上性能良好,但是在处理大规模问题时效率表现不一致. 实验发现,增强爬山算法在其中的一些问题上求解失败,随后被调用的贪婪最好优先算法求解效率较低,因此 FF 的整体效率降低. 其原因在于,经 IPP 方法处理的 ADL 动作包含组件诱发关系,该关系的处理需要基于动作的删除效果进行推理,然而,松弛图规划完全忽略了动作

的删除效果,无法进行必要的推理,提供了不完备的有利动作,导致增强爬山算法失败.

我们以 ADL 语言描述的一个简单规划任务为例(见图 2)说明基于松弛图规划的“有利动作”策略的失效情况. 图 2 所示的规划任务 P 涉及两个动作 a_0 和 a_1 . 其中 a_0 的前提为命题 p , 带有两个效果: r 是一个无条件效果,表示 a_0 执行后命题 r 为真; (when $q \rightarrow p$) 是一个条件效果,表示如果命题 q 在 a_0 执行前为真,则 a_0 执行后命题 p 为假. 在初始状态中,命题 p 和 q 均为真,目标条件是 p 和 r 为真. 分枝扩展法将每个动作转化一个组件集合,其中每个组件对应于该动作的一个(条件)效果. 组件的形式为 $e = (con(e), add(e), del(e))$, 其中 $con(e)$, $add(e)$ 和 $del(e)$ 均是原子命题的集合,分别表示组件 e 的发生条件、添加效果和删除效果. 由于动作 a_0 包含一个无条件效果 r 和一个条件效果 (when $q \rightarrow p$), 所以在图 3 中 a_0 包含两个对应的组件 e_0 和 e_1 .

对于规划任务 P , 增强爬山算法从初始状态 I 开始搜索,首先调用松弛图规划计算 I 的目标距离估计. 松弛图规划根据 I 建立的规划图如图 4 所示(注意:组件的删除效果被忽略),其中的虚线表示 Noop 动作. 本文假定组件规划图从时间步 0 开始构

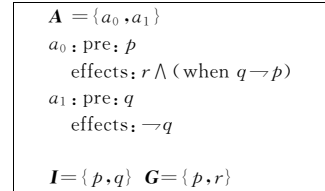


图 2 一个规划任务 P

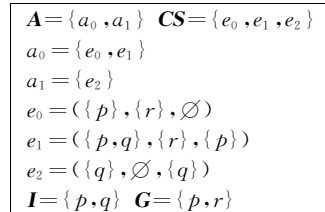


图 3 分枝扩展法处理 P 的结果

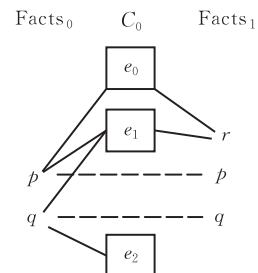


图 4 状态 I 对应的松弛规划图

造,每个时间步包括一个命题列和一个组件列.第 0 命题列包含命题 p 和 q ,第 0 组件列包含组件 e_0, e_1 和 e_2 ,第 1 命题列包含命题 r, p 和 q .由于所有的目标命题均在图中出现,图扩展过程结束.松弛图规划算法根据 Noop 动作优先原则以及具有最小难度的动作优先原则^[4],选择组件 e_0 添加命题 r ,选择命题 p 对应的 Noop 动作添加命题 p ,最终得到一条松弛规划 $\langle a_0 \rangle$,并反馈给增强爬山模块如下信息:当前状态到达目标状态的距离是 1,有利动作集合为 $\{a_0\}$.增强爬山算法应用动作 a_0 转移到状态 $s = \{q, r\}$.由于该状态不满足目标条件,增强爬山算法再次调用松弛图规划计算 s 的目标距离估计和有利动作集合.松弛图规划根据 s 建立的松弛规划图达到稳定却不包含目标条件 p ,因此,反馈给增强爬山算法如下信息:当前状态到达目标状态的距离为 ∞ .随后,增强爬山算法报告搜索失败.

事实上,规划任务 P 存在规划解:从初始状态出发,只要依次执行动作 a_1 和 a_0 就可以到达一个目标状态.松弛图规划无法发现有利动作 a_1 的原因在于未考虑动作 a_0 中组件 e_0 对组件 e_1 的诱发(induce)关系(由于 e_0 和 e_1 均是动作 a_0 的一部分,当 a_0 执行时, e_0 和 e_1 都可能影响执行结果).如果它能够预测到在初始状态 I 上执行 a_0 时 e_1 将发生并且删除已经实现的目标命题 p ,从而在动作 a_0 执行之前选择动作 a_1 来破坏组件 e_1 的发生条件,则能够成功地引导增强爬山算法.然而,忽略动作的全部删除效果使得这种推理变得不可能,“有利动作”策略不可避免地发生失误.

3 朴素组件规划图上的延迟部分推理

“有利动作”剪枝策略是 FF 的关键技术,能够以指数级别降低搜索空间的规模.避免该策略发生失误的最简单方式是采用全扩展法处理 ADL 动作,但是需要巨大的存储空间.当采用 IPP 方法或分枝扩展法时,避免该策略发生失误的方法比较复杂.其中的困难在于规划算法需要估计在后续规划过程中可能发生的所有组件诱发关系,并保留对诱发关系的消极作用进行抵制的能力^[19].但是,预测所有可能的组件诱发关系需要较高的计算代价,而且这些诱发关系可能分布在多个规划解之中,而启发函数只需要估计其中一条规划解的长度.基于以上分析,我们定义了与单个松弛规划解相关的组件诱发关系——“松弛诱发”,在构造规划图过程中保

留动作的删除效果,在规划解提取过程中判断组件之间的松弛诱发关系并进行相应的抵制.下面,我们首先介绍处理条件效果的 3 种方法,说明采用 IPP 方法或者分枝扩展法的必要性,之后提出基于分枝扩展法的朴素组件规划图及延迟部分推理方法.

3.1 条件效果处理方法比较

和 STRIPS 语言相比,ADL 语言具有更强的知识表达能力,可以描述更为复杂的世界模型.其特点在于允许动作模型包含全称量词和条件效果.目前处理条件效果的方法主要有 3 种:(1)全扩展方法;(2)IPP 方法;(3)Weld 等人提出的分枝扩展法.全扩展法将一个带有条件效果和全称量词的 ADL 动作转化为多个独立的 STRIPS 动作,即基于全扩展法的规划算法在状态上应用一个动作时不必考虑其它动作对状态转移的影响.但是,该方法的缺点在于转化后的动作数目随着原问题包含的对象数目和动作效果数目呈指数级别增长,需要大量的存储空间.IPP 方法和分枝扩展法均采用部分处理的方式,将一个 ADL 动作的不同效果分离,存储空间需求随着原问题涉及的对象数目和动作效果数目呈线性级别增长.在本文,我们采用分枝扩展法.分枝扩展的主要思想是将动作所有的效果条件化,根据每个效果构造一个动作组件(component),一个组件由三部分构成:发生条件、添加效果和删除效果.组件的发生条件是被转化动作的前提和对应的条件效果前提的合取,组件的添加效果和删除效果根据对应的条件效果生成.例如,在图 2 的规划任务中动作 a_0 被分枝扩展法处理为如图 3 所示的两个组件 e_0 和 e_1 .我们一般称组件 e_0 和 e_1 来自(属于)动作 a_0 .

IPP 方法和分枝扩展方法在显著减少动作数目的同时给规划算法带来额外的困难,其主要原因是组件诱发关系的存在^[16].以图规划方法为例.规划图的扩展算法基于组件进行推理,而不是基于动作的推理;但最终的规划解由动作构成,而不是由组件构成.这使得在规划图构造过程中,需要使用复杂的规则计算组件间可能存在的互斥关系;在规划解提取过程中,需要做出抵制决策(confrontation)来避免组件之间由于相互诱发而产生的消极作用,此类回溯点的增加降低了回溯算法的效率.

3.2 朴素组件规划图

组件规划图由 Weld 等人提出,用于扩展图规划方法处理 ADL 规划问题的能力^[16].它的构造方法和规划图的构造方法类似,区别在于前者由命题列和组件列交替构成.本文假定组件规划图从时间

步 0 开始构造, 每个时间步包含一个命题列和一个组件列. 对于组件 c , 当它的所有发生条件都在第 i 命题列出现并且不存在两两互斥时被加入第 i 组件列, 其添加命题被加入第 $i+1$ 命题列. 在组件规划图的扩展过程中计算如下的互斥关系和组件诱发关系^[16].

定义 1. 设 c_n 和 c_m 是两个组件, 如果 (1) c_n 和 c_m 来自两个不同的动作, 并且组件 c_n 删除组件 c_m 的发生条件或者添加效果, 或者 (2) c_n 的某个前提和 c_m 的某个前提在第 i 命题列互斥, 或者 (3) 存在第 3 个组件 c_k , c_k 由 c_n 在第 i 列诱发, 且 c_k 和 c_m 互斥, 则称组件 c_n 和 c_m 在第 i 组件列 (第 i 时间步) 互斥.

其中由 (1)、(2) 和 (3) 引起的三类互斥关系分别称为干扰效果 (interference effects)、竞争需要 (competing needs) 和诱发组件 (induced components). 诱发的定义如下.

定义 2. 设 c_n 和 c_k 是两个组件, 如果 (1) c_n 和 c_k 来自同一个动作, 并且 (2) c_n 和 c_k 不互斥, 并且 (3) $con(c_k)$ 中的每个命题的否定在第 i 命题列不被满足, 则称组件 c_n 在第 i 组件列 (第 i 时间步) 诱发组件 c_k .

组件 c_n 诱发组件 c_k 意味着 c_n 的执行必然伴随 c_k 的执行. 我们简要介绍组件诱发的含义. 设组件 c_n 和 c_k 均来自动作 a , 且动作 a 执行前的状态为 s , $con(c_n)$ 和 $con(c_k)$ 分别表示 c_n 和 c_k 的发生条件. c_n 诱发 c_k 的情况可以分为两种: 一种情况为, 当 $con(c_k) \subseteq con(c_n)$ 时 c_n 必然诱发 c_k ; 另一种情况为, 当 $con(c_n) \subset con(c_k)$ 且 $con(c_n) \subseteq s$ 时 c_n 条件诱发 c_k . 称后一种诱发是条件的, 因为诱发的发生与动作 a 执行前的状态 s 相关. 如果存在命题 $p \in con(c_k)$ 且 $p \notin s$, 则 c_n 的发生不会诱发 c_k 的发生. 在规划过程中, 如果发现 c_n 诱发 c_k 但不希望 c_k 发生, 则有两种选择: 当 c_n 必然诱发 c_k 时, 可以选择放弃 c_n ; 当 c_n 条件诱发 c_k 时, 可以尝试抵制 c_k 的发生. 抵制组件 c_k 发生的简单方法是使某个命题 $p \in (con(c_k) - con(c_n))$ 在动作 a 执行之前为假, 即选择一个在 s 中可应用的动作 a' 删除命题 p 而不阻碍组件 c_n 的发生. 我们通常称 a' 这类动作为抵制动作. 抵制过程是基于因果链接的规划方法的重要部分^[8-9].

定义 1 中的 3 种互斥关系在组件规划图的扩展阶段计算, 反映规划解中动作和命题的相互影响. 基于忽略动作删除效果的假定, FF 系统的松弛图规划不需要计算任何互斥关系. 然而正如在本文的第 2

节所述, 完全忽略这 3 种互斥关系将导致“有利动作”策略失效, 从而引起增强爬山算法的失败. 避免这种失败的一个方案是考虑动作的删除效果并在规划图构造的过程中计算上述 3 种互斥关系. 然而, 互斥关系的计算和检查需要大量的计算时间^[19], 而在提取规划解的过程中进行组件抵制的尝试也可能导致大量的回溯^[16]. 为了设计一个能够产生有效的启发式信息而且计算费用相对较低的推理方法, 我们对定义 1 中的 3 种互斥关系进一步划分. 显然, 前两种互斥关系在 STRIPS 和 ADL 规划中均存在, 我们称为必然互斥, 第 3 种互斥关系仅在 ADL 规划中存在, 我们称为条件互斥. 在 STRIPS 规划问题中, 松弛图规划虽然完全忽略了必然互斥关系, 但是 FF 在几乎所有的规划任务上表现优异. 基于这个观察, 我们希望新设计的计算松弛规划解的方法能够在 STRIPS 域上与松弛图规划具有相同的特点. 因此, 我们提出的方法仅考虑诱发组件型互斥关系.

为了计算诱发组件型互斥关系, 我们构造一个简化的组件规划图——朴素组件规划图 (Naïve Components Planning Graph, NCPG). 朴素组件规划图是一种不包含任何互斥关系的组件规划图, 其中每个动作都保留删除效果 (图 5 显示了第 2 节中规划任务 P 的初始状态对应的朴素组件规划图, 其中点划线指向组件的删除命题). 当朴素组件规划图构造完毕后, 我们使用下面提出的延迟部分推理方法考虑诱发组件型互斥关系, 计算松弛规划解. 基于朴素组件规划图的延迟部分推理方法 (Delayed Partly Reasoning on a Naïve Components Planning Graph, DPR-NCPG) 与基于松弛规划任务的图规划方法 (松弛图规划) 主要存在两点差别: (1) 前者基于原始的规划任务, 采用简化的计算过程, 而后者基于简化的规划任务, 采用标准的计算过程. (2) 前者考虑一部分互斥关系, 后者忽略全部互斥关系. 下面介绍 DPR-NCPG 计算松弛规划解的过程.

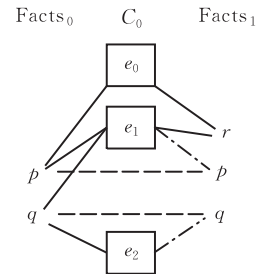


图 5 状态 I 对应的朴素组件规划图

3.3 延迟部分推理

延迟部分推理方法是一种与图规划类似的回溯式搜索算法,在搜索的过程中计算一部分组件互斥关系.延迟部分推理方法计算的组件互斥关系通过“松弛诱发”和“松弛互斥”概念刻画.首先提出两个限定:(1)在朴素组件规划图上计算规划解的过程中假定组件所属于的动作的执行顺序就是组件被选择的顺序;(2)在规划求解的过程中仅考虑在已定的动作执行顺序下存在的诱发组件互斥关系.

定义 3. 在延迟部分推理的求解过程中,设 c_n 和 c_k 为第 i 组件列属于同一个动作的两个组件,称 c_n 松弛诱发 c_k ,如果 $con(c_n) \subset con(c_k)$ 且 $\forall p \in con(c_k)$ 满足下列条件之一:(1) p 被 i 列已经选择的组件添加;(2) p 为 j 列 ($j \leq i$) 的目标;(3) $p \in con(c_n)$;(4) $i=0$,并且 p 出现在第 0 命题列.

延迟部分推理方法在选择一个组件 c_n 之前检查该组件与已选择组件的松弛互斥关系并进行相应的处理.

定义 4. 在延迟部分推理的求解过程中,设第 i 组件列已被选择的组件序列为 $SC_i = \langle c_1, c_2, \dots, c_n \rangle$,称组件 c_k 与 SC_i 存在松弛互斥关系,如果组件 c_n 松弛诱发组件 c_k ,并且 c_k 删除第 $i+1$ 命题列已被添加的目标或者删除第 j ($j \leq i$) 命题列的目标.

松弛互斥关系与求解过程相关,随着规划解的变化而变化.它预测单个规划解中可能存在的组件诱发关系及其消极作用.当发现组件 c_k 与 SC_i 存在松弛互斥时,延迟部分推理方法尝试对 c_k 进行抵制以避免其消极作用.抵制的方法为:选择某个命题 $p \in (con(c_k) - con(c_n))$ 且 p 不是(子)目标,之后选择一个第 j ($j \leq i$) 列的组件删除 p .

```

function PInduce( $c_n, c_k, i$ )
1. for each fact  $p \in con(c_k)$ 
2.   if  $p$  is not true at facts level  $i+1$ 
3.     if  $p$  is not a goal at facts level  $j$  ( $j \leq i$ )
4.       if  $p \notin con(c_n)$ 
5.         if  $i=0$  and  $p$  is not at facts level 0
6.           return FALSE
7. return TRUE
procedure ConfrontCheck( $c_n, i$ )
1. Confronted=FALSE
2. let  $C = \{c_k | c_k \text{ and } c_n \text{ belong to the same action } a, \text{ and } con(c_n) \subset con(c_k)\}$ 
3. for each  $c_k \in C$ 
4.   if PInduce( $c_n, c_k, i$ )
5.     if  $c_k$  deletes a goal  $g$ , which is true at facts level  $i+1$  or appears at  $j \leq i$ 
6.       Confronted=FALSE
7.       for each fact  $q \in con(c_k) - con(c_n)$  and !Confronted
8.         for each component  $c_m$ , s. t.,  $level(c_m) \leq i$ , such that  $q \in del(c_m)$ 
9.           if not DeleteGoal( $c_m, i$ )
10.            add action  $a'$  to which  $c_m$  belongs, into the current relaxed plan
11.            update the heuristic estimation
12.            if ( $i=0$ )
13.              add  $a'$  to the helpful actions set
14.            Confronted=TRUE
15. break

```

图 6 DPR-NCPG 方法的两个模块:松弛诱发判断和抵制检查

图 6 给出 DPR-NCPG 中实现定义 3 和 4 以及进行抵制处理的算法. DPR-NCPG 的整体过程类似于图规划的回溯算法. DPR-NCPG 为第 $i+1$ 命题列的一个目标命题 g 寻找支持组件时,如果 g 存在于某个动作组件的添加效果,则该组件成为备选组件.仍然使用 Noop 优先原则和具有最小难度的动作优先原则为目标 g 在多个备选组件中选择一个支持组件.若一个组件被选定,则这个组件所来自的动作被加入最终的松弛规划解.如前所述,我们认为规划解的多个动作是串行执行的,假定动作的执行顺序是它们被选择的顺序.因此,当第 i 组件列的组

件 c_n 被选择以支持第 $i+1$ 列的目标命题 g 时,如果某个命题 $p \in con(c_n)$ 被时间步 i 已经选择的组件添加,则不再将 p 作为第 i 命题列的目标.在一个选定的组件 c_n 加入第 i 组件列已经选择的支持组件序列 SC_i 之前,调用过程 ConfrontCheck() 对 c_n 进行松弛诱发关系检查和处理. ConfrontCheck() 过程逐个考察与 c_n 来自同一个动作的每个组件 c_k . 首先调用函数 PInduce() 判断在当前的组件执行顺序下组件 c_n 是否松弛诱发组件 c_k . 若 c_n 诱发 c_k , 并且 c_k 和 SC_i 存在松弛互斥关系,则对 c_k 进行抵制:选择某个命题 $p \in (con(c_k) - con(c_n))$, 对于命题 p , 选择一个删除

p 但不删除时间步 $i+1$ 已经实现目标的组件 c_m , 若不存在这样的 c_m , 则放弃抵制 c_k . 若存在这样的 c_m , 则将 c_m 加入 SC_i , 将 c_m 的前提作为第 i 命题列的子目标向前传递. 此过程的一个重要步骤为: 若 c_m 在时间步 0 的组件列, 则将 c_m 所属于的动作加入有利动作集.

下面以图 5 的朴素组件规划图为例介绍 DPR-NCPG 方法计算松弛规划解的大致过程. 当 DPR-NCPG 为时间步 1 的目标 r 选定支持组件 e_0 后, 在将 e_0 加入已选择组件集 SC_0 之前, 调用 *ConfrontCheck()* 对 e_0 进行处理. 由于组件 e_1 和 e_0 来自同一个动作 a , 处理过程首先考察组件 e_1 . 组件 e_1 由组件 e_0 松弛诱发而且 e_1 删除第 0 列的目标 p , 因此 e_1 与 SC_0 松弛互斥. *ConfrontCheck()* 算法选择 e_2 删除命题 q 以避免 e_1 的消极作用, 将 e_2 加入 SC_0 , 同时将 e_2 所属于的动作 a_1 加入有利动作集. 由于动作 a_0 不存在其它需要检查的组件, *ConfrontCheck()* 过程结束. DPR-NCPG 将组件 e_0 加入 SC_0 . 最终, DPR-NCPG 计算的松弛规划为 $\langle a_1, a_0 \rangle$, 有利动作集合为 $\{a_1, a_0\}$. 容易看出, 增强爬山算法得到由 DPR-NCPG 反馈的有利动作集合后, 能够扩展到存在解的状态空间, 实现成功求解. 相比之下, 松弛图规划算法无法发现有利动作 a_1 .

与松弛图规划相比, DPR-NCPG 考虑组件之间的动态互斥关系, 最终获得的启发式函数所含的信息量更大. 另一方面, 朴素组件规划图的构造过程不需要计算互斥关系, 延迟部分推理方法仅考虑一部分互斥关系, 因此 DPR-NCPG 的计算代价较低. 从剪枝策略角度看, 松弛诱发关系和松弛互斥关系的判断随着松弛规划解的不同而变化, 能够预测后续规划过程中松弛诱发关系产生的消极影响; 抵制处理算法为后续规划保留了处理该类消极影响的动作. 因此, 基于 DPR-NCPG 的启发式函数和“有利动作”策略都具有更好的性能.

4 实验结果与分析

我们以 FF 的最新版本 FF-v2.3^① 为基础, 设计了基于 DPR-NCPG 方法的规划系统 FFc. 本节将从规划效率和规划质量(规划解长度)两个方面比较 FF 和 FFc. 实验在一台配置为奔腾 IV 2.5GHz 处理器, 256MB 内存, 运行 Linux 操作系统的机器上进行. 我们使用国际通用的 ADL 规划域: Briefcase world、Schedule 和 Elevator, 并对实验结果进行了

初步分析. 在以下各图表中, “EHC”代表增强爬山算法, “SL”是解长度 (Solution Length) 的缩写, “SS”代表访问的状态数目, “TM”代表花费的总时间, “US”代表无解.

4.1 Briefcase world 域

Briefcase world 域是一个经典的 ADL 规划域. 该域使用一个公文包将物品从其初始地运送到其目的地. 物品可以放入 (put-in) 公文包中也可以从公文包中取出 (take-out). 当使用 move 动作移动公文包时, 放于公文包内的物品伴随着公文包移动 (条件效果). 在 Briefcase world 域进行规划的难点在于处理 move 动作的条件效果并判断其引发的消极作用, 规划算法需要在合适的时间使用 take-out 动作避免条件效果的消极作用. 以该域的一个简单问题为例. 初始状态为物品 o_1 和 o_2 在公文包中并且公文包在 L 地, 物品 o_1 的目的地为地点 L , 物品 o_2 的目的地为地点 L' . 从直观上看, o_1 已经处于目的地, 不需要任何处理; 仅需要将公文包从 L 地移动到 L' 地来运送 o_2 . 然而, 在移动公文包之前, 如果未将 o_1 从公文包中取出, 则 o_1 将随着公文包被运送到 L' 地 (move 动作的条件效果). 如果规划算法希望在运送 o_2 时不影响 o_1 的位置, 则必须在移动公文包之前采用一个 take-out 动作将 o_1 从公文包中取出. 在该领域的大多数问题实例上, FF 系统的 rGP 模块无法计算条件效果产生的消极影响以及避免该消极影响的动作, 产生的“有利动作”策略总是丢弃 take-out 动作, 从而引起增强爬山算法的失败, 随后的求解过程主要由贪婪最好优先算法完成, 耗时间较长. 相对而言, FFc 系统的 DPR-NCPG 引导增强爬山算法在所有的测试问题上求解成功, 表现出较好的性能. 本文使用 FF 系统提供的问题生成器^②生成随机的测试问题. 表 1 显示了 FF 与 FFc 的对比实验结果,

表 1 Briefcase world 域实验对比

问题	FF 实验结果			FFc 实验结果		
	SL	SS	TM/s	SL	SS	TM/s
BRF5	13	74	0.00	15	21	0.15
BRF10	33	435	0.62	42	201	0.31
BRF15	47	1238	7.66	63	437	3.60
BRF20	59	1976	89.85	81	895	66.72
BRF25	79	3600	492.97	88	549	90.62
BRF26	81	2889	428.91	90	570	108.74
BRF27	85	2958	457.66	97	756	160.00
BRF28	90	4130	754.21	100	1243	347.50
BRF29	86	5196	1160.40	105	1558	417.66
BRF30	95	4349	1085.40	120	2412	723.91

① <http://members.deri.at/~joergh/ff.html>
② <http://members.deri.at/~joergh/ff-domains.html>

从中可以看出,在小规模问题上 FFc 的效率优势不明显,但是随着问题规模的增加,FFc 的效率比 FF 具有成倍的提升.

4.2 Schedule 域

Schedule 域也是一个由 ADL 语言描述的规划域,其中的动作包含条件效果.该域建模一类加工器件的生产调度问题,即使用多个车床加工一定数量的器件,可以采用抛光(polish)、打孔(punch hole)、碾压(roll)和涂漆(painting)等操作对器件进行处理. Schedule 域是一个较难处理的规划域^[12]. 本文采用的测试问题取自 AIPS2000 规划竞赛.在多数问题的求解过程中,启发式函数涉及的松弛规划任务包含大量的并行动作,松弛规划图一般只需要扩展 3 个时间步,因此,FF 系统的启发函数对增强爬山算法的引导能力不强,而“有利动作”策略成为 FF 在此域高效求解的主要因素(仅有约 2%的可用动作被标记为有利动作)^[4]. 在 FFc 系统中,延迟部分推理方法由于考虑一部分组件诱发关系提高了启发函数和剪枝策略的性能.图 7 和图 8 分别显示了 FF 和 FFc 在规划效率和规划质量方面的对比.结果表明,无论在 rGP 的引导下还是在 DPR-NCPG 的引导下,增强爬山搜索在几乎所有的问题上都能够成

功.但是如图 7 和图 8 所示,在大多数问题中,增强爬山算法在 DPR-NCPG 引导下耗费的时间较少,而且可以获得较好的规划解.

4.3 Miconic-ADL 域

Miconic-ADL 域建模一类使用电梯运送乘客的问题,其中的动作包含复杂的前提和条件效果.运送乘客时,需要考虑多种约束,例如,必须首先为 VIP 乘客服务,在运送某些乘客的过程中电梯不能停止,某些乘客需要服务员陪同等等.当电梯停在某一楼层时(stop 动作),所有目的地为该楼层的乘客走出电梯,所有在该楼层等候的乘客进入电梯(条件效果).显然,电梯移动时,其中的乘客将随着电梯移动,类似于 Briefcase world 域中物品随公文包移动的现象,但是该域与 Briefcase world 域的差别在于电梯中的乘客在电梯停止时自动地走出电梯,规划算法不需要采用某个动作来达到这个结果,因此基于 rGP 的“有利动作”策略较少引发增强爬山算法的失败.在 Miconic-ADL 域,我们使用 AIPS2000 的测试问题集.实验数据见表 2.表 2 标有“EHC”的列标记增强爬山算法是否搜索成功,“T”代表搜索成功,“F”代表搜索失败.在 FF 系统中,由 rGP 引导的增强爬山算法在该域的一些问题上求解失败.实验结果表明,DPR-NCPG 和 rGP 对增强爬山算法的影响不同.从结果可以看出,在一些问题上(如问题 f12-1 和 f14-2)增强爬山算法在 rGP 引导下失败,在 DPR-NCPG 引导下取得成功;当增强爬山算法在 rGP 引导下搜索成功时,在 DPR-NCPG 引导下仍然搜索成功.

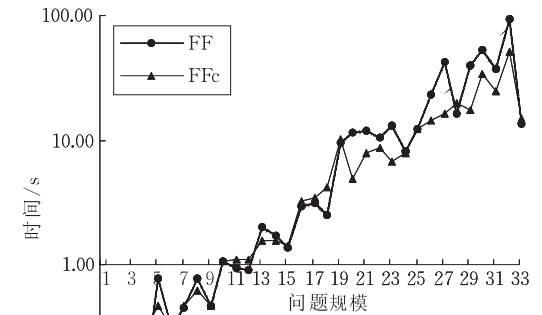


图 7 Schedule 域规划效率对比(时间轴为对数坐标)

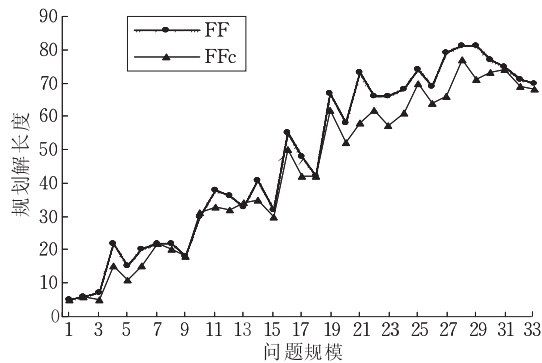


图 8 Schedule 域规划质量对比

表 2 Miconic-ADL 域实验对比

问题	FF 实验结果				FFc 实验结果			
	SL	SS	TM	EHC	SL	SS	TM	EHC
f10-0	34	10376	13.43	F	35	594	1.41	F
f10-1	33	92	0.62	T	32	134	0.78	T
f10-2	US				US			
f10-3	34	13782	17.19	F	34	5533	7.82	F
f10-4	35	103	0.63	T	31	69	0.62	T
f11-0	34	71	0.62	T	33	68	0.93	T
f11-1	37	132	0.93	T	38	138	0.78	T
f11-2	33	68	0.63	T	32	68	0.78	T
f11-3	32	848	2.03	F	32	535	1.41	F
f11-4	36	148	0.94	T	35	98	0.78	T
f12-0	43	104	0.94	T	43	109	1.10	T
f12-1	39	1350	4.21	F	43	176	1.25	T
f12-2	42	86	0.94	T	42	86	1.10	T
f12-3	38	579	2.03	F	40	3842	9.37	F
f12-4	39	77	0.94	T	39	77	1.09	T
f13-0	41	97	1.10	T	42	106	1.25	T
f13-1	39	133	1.26	T	39	130	1.41	T
f13-2	38	573	2.18	F	40	623	2.35	F
f13-3	48	160	1.41	T	46	115	1.25	T

(续 表)

问题	FF 实验结果				FFc 实验结果			
	SL	SS	TM	EHC	SL	SS	TM	EHC
f13-4	38	6703	12.97	F	40	23852	47.03	F
f14-0	42	633	2.97	F	42	611	3.12	F
f14-1	40	155	1.72	T	48	325	2.34	T
f14-2	43	683	3.12	F	47	163	1.56	T
f14-3	49	178	1.71	T	46	152	1.56	T
f14-4	47	112	1.41	T	48	113	1.56	T

5 结 语

FF 作为一个启发式搜索方法应用于规划问题的典型系统,自从出现以来,一直受到研究者的关注. Hoffmann 对松弛图规划和增强爬山算法结合的方法从实验角度和理论角度给予了深入的分析. FF 最突出的特点是使用松弛图规划产生的规划解的长度作为估计值,使用“有利动作”策略裁剪搜索空间. 其它的研究者从这两个方面借鉴 FF 的思想并提出多种改进的方法. 如 Helmert 针对松弛图规划忽略动作删除效果时估计偏差过大的不足,提出了基于因果图分析(casual graph analysis)的启发式函数和称为“有利转移”(helpful transitions)的剪枝策略^[6],设计的 Fast Downward 规划系统在 STRIPS 规划域的性能优于 FF. 但是, Fast Downward 在 Schedule 域的效率明显低于 FF,其中的原因有待进一步研究. 此外,“有利转移”策略通常存在过度剪枝的情况; Vidal 提出的补救动作(rescue actions)和前瞻(look ahead)方法^[19],既弥补了松弛图规划提供不完全有利动作的不足又利用了松弛图规划解的信息,提高了规划求解效率. 然而,该方法存在规划解质量较低的不足;姜云飞等人从提高启发函数的信息量的角度,通过计算经典的组件规划图,提高 FF 识别僵局(dead end)的能力^[20],然而,构造经典组件规划图需要大量的计算代价而且无法预测所有的死点,此外,有利动作的不完全问题仍然存在.

据我们所知,研究者尚未发现 FF 系统中基于松弛图规划的“有利动作”策略失效的主要原因. 本文的研究指出,在 FF 使用分枝扩展法处理条件效果的同时,松弛图规划完全忽略了分枝扩展法所带来的复杂性,在必须处理诱发组件互斥关系的规划任务上无法发现使增强爬山算法扩展到解空间的有利动作,进而丧失了提供有效引导信息的作用,导致 FF 的效率急剧降低. 针对这一问题,本文提出在朴素组件规划图上进行延迟部分推理的方法. 初步的实验结果表明,该方法是在计算代价和信息量之间

的一个较好权衡,提高系统效率的作用比较明显. 在组件规划图中,诱发组件互斥反映动作之间复杂的相互影响. 由于计算所有诱发组件互斥关系的代价过高,我们将继续研究如何选择性地计算互斥以及如何近似地处理互斥的更高效方法,进而研究 ADL 语言在复杂规划模型中的处理方法^[21-24].

参 考 文 献

[1] Kautz H, Selman B. Pushing the envelope: Planning, propositional logic, and stochastic search//Proceedings of the 13th National Conference on Artificial Intelligence. Portland, Oregon, 1996: 1194-1201

[2] van den Briel M, Kambhampati S. Optiplan: Unifying ip-based and graph-based planning. Journal of Artificial Intelligence Research, 2005, 24: 919-931

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

[4] Hoffmann J, Nebel B. The ff planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligent Research, 2001, 14: 253-302

[5] Gerevini A, Serina I, Saetti A et al. Local search techniques for temporal planning in LPG//Proceedings of the 13th International Conference on Automated Planning and Scheduling. Trento, Italy, 2003: 62-71

[6] Helmert M. The fast downward planning system. Journal of Artificial Intelligence Research, 2006, 26: 191-246

[7] Chen Y, Hsu C, Wah B. Temporal panning using subgoal partitioning and resolution in sgplan. Journal of Artificial Intelligence Research, 2006, 26: 323-369

[8] Weld D. An introduction to least-commitment planning. AI Magazine, 1994, 15(4): 27-61

[9] Vidal V, Geffner H. Branching and pruning: An optimal temporal POCL planner based on constraint programming. Artificial Intelligence, 2006, 170(3): 298-335

[10] Fikes R, Nilsson N. Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 1971, 2(3): 189-203

[11] Russell S, Norvig P. Artificial Intelligence: A Modern Approach. Englewood Cliffs: Prentice Hall, 2003

[12] Hoffmann J. Where ignoring delete lists works: Local search topology in planning benchmarks. Journal of Artificial Intelligence Research, 2005, 24: 685-758

[13] Pednault E. Adl: Exploring the middle ground between strips and the situation calculus//Proceedings of the International Conference on AI Planning Systems. Tornoto, Canada, 1996: 142-149

[14] Gazen B, Knoblock C. Combining the expressivity of UCPOP with the efficiency of graphplan//Proceedings of the 4th European Conference on Planning. Toulouse, France, 1997: 221-233

- [15] Koehler J, Nebel B, Hoffmann J, Dimopoulos Y. Extending planning-graphs to an adl subset//Proceedings of the 4th European Conference on Planning. Toulouse, France, 1997: 273-285
- [16] Anderson C R, Smith D E, Weld D S. Conditional effects in graphplan//Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems. Pittsburgh, Pennsylvania, USA, 1998: 44-53
- [17] Koehler J, Hoffmann J. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. Journal of Artificial Intelligence Research, 2000, 12: 338-386
- [18] Blum A, Furst M. Fast planning through planning graph analysis. Artificial Intelligence, 1997, 90: 281-300
- [19] Vidal V. A lookahead strategy for heuristic search planning//Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04). Whistler, British Columbia, Canada, 2004: 150-160
- [20] Zhu Man-Fei, Jiang Yun-Fei. CeFF: A method handling conditional effects based on components. Chinese Journal of Computers, 2004, 27(12): 1601-1611(in Chinese)
(朱曼菲, 姜云飞. CEFF——一种基于组件的条件效果处理方法. 计算机学报, 2004, 27(12): 1601-1611)
- [21] Zhou Jun-Ping, Yin Ming-Hao, Gu Wen-Xiang, Sun Ji-Gui. Decrease observation variables for strong planning. Journal of Software, to appear(in Chinese)
(周俊萍, 殷明浩, 谷文祥, 孙吉贵. 部分可观察强规划中约减观察变量的研究. 软件学报, 待发表)
- [22] Yin Ming-Hao, Sun Ji-Gui, Lu Shuai, Cai Dun-Bo. A novel framework for Plan recognition: Graphplan as a basis//Proceedings of the 7th IJCAI International Workshop on Nonmonotonic Reasoning, Action and Change. Hyderabad, India, 2007
- [23] Sun Ji-Gui, Yin Ming-Hao. Recognizing agent's intention incrementally: Graphplan as a basis. Journal of Frontiers of Computer Science in China, 2007, 1: 26-36
- [24] Yin Ming-Hao, Lin Hai, Sun Ji-Gui. Counting models using extension rules//Proceedings of the 22nd AAAI Conference on Artificial Intelligence. Vancouver, British Columbia, Canada, 2007: 1916-1917



CAI Dun-Bo, born in 1981, Ph. D. candidate. His main research interests include AI planning, constraint reasoning and learning in planning.

YIN Ming-Hao, born in 1979, Ph. D. candidate. His main research interests include AI planning, plan recognition, Fuzzy reasoning and automated reasoning.

GU Wen-Xiang, born in 1947, professor, Ph. D. supervisor. His main research interests include AI planning and Fuzzy reasoning.

SUN Ji-Gui, born in 1962, professor, Ph. D. supervisor. His main research interests include automated reasoning, AI planning and constraint programming.

LIU Ke-Cheng, born in 1980, master. His main research interests include AI planning and plan recognition.

Background

Artificial Intelligence planning is one of the most active topics in AI research. The aim of the group is to make AI planning methods applicable in realistic problems, specifically, intends to: (1) propose more efficient algorithms for AI planning, (2) make AI planning representation methods richer, (3) make AI planning method applicable under uncertainty environment.

They contributions to this topic now include: (1) a plan recognition algorithm that can be regarded as the counterpart of the famous classical planning system Graphplan, (2) a human-machine collaboration method that integrates plan recognition technology, (3) a causal graph based heuristic approach for conformant planning, (4) a system JLU-CD that

is competitive to state-of-the-art conformant planners, (5) a planner JLU-LAO that can solve un-deterministic planning problems, (6) an efficient SAT solving algorithm based on extension rules, (7) an efficient model counting algorithm based on extension rules.

This paper focuses on planning in ADL, which is an expressive planning domain description language. A method called DPR-NCPG is introduced to deal with conditional effects for state-space planning. Experimental results show the power of the proposed method. The work is helpful in improving the efficiency of state-space based conformant and contingent planning methods.