软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(7):3099−3115 [doi: 10.13328/j.cnki.jos.006859]

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

智能规划中面向简单偏好的高效求解方法*

陆 旭[1,2], 于 斌[1,2], 段振华[1,2], 王德奎[3], 陈 矗[4], 崔 进[5]

[1](西安电子科技大学 计算机科学与技术学院, 陕西 西安 710071)

[2](综合业务网理论及关键技术国家重点实验室(西安电子科技大学), 陕西 西安 710071)

[3](西北大学 信息科学与技术学院, 陕西 西安 710127)

[4](曲阜师范大学 计算机学院, 山东 曲阜 273165)

[5](西安石油大学 计算机学院, 陕西 西安 710065)

通信作者: 于斌, E-mail: byu@xidian.edu.cn; 段振华 E-mail: zhhduan@mail.xidian.edu.cn

摘 要: 智能规划(AI planning)简称规划, 是人工智能领域的一个重要分支, 在各领域均有广泛应用, 如工厂车间作业调度、物资运输调度、机器人动作规划以及航空航天任务规划等. 传统智能规划要求规划解(动作序列)必须最终实现整个目标集合, 这种目标一般被称为硬目标(hard goal). 然而, 许多实际问题中, 求解的重点并不只是尽快实现目标以及尽量减少动作序列产生的代价, 还需考虑其他因素, 如资源消耗或时间约束等. 为此, 简单偏好(也称软目标 soft goal)的概念应运而生. 与硬目标相反, 简单偏好是可以违背的. 本质上, 简单偏好用于衡量规划解质量的优劣, 而不会影响规划解是否存在. 现有关于简单偏好的研究进展缓慢, 在规划解质量方面不尽如人意, 即求得的规划解与最优解的差距较大. 提出了一种求解简单偏好的高效规划方法, 将简单偏好表达为经典规划(classical planning)模型的一部分, 并利用 SMT (satisfiability modulo theories)求解器识别多个简单偏好之间的各种关系, 从而约简简单偏好集, 减轻规划器的求解负担. 该方法的主要优势在于: 一方面, 提前对简单偏好集进行裁剪, 在一定程度上缩减搜索的状态空间; 另一方面, 直接利用现有高效经典规划器进行求解, 而无须提出专用的规划算法, 可扩展性较强. 基准规划问题的实验结果表明, 该方法在提升规划解质量方面表现优异, 尤其适用于简单偏好之间不是相互独立的情况.

关键词: 智能规划; 简单偏好; SMT

中图法分类号: TP18

中文引用格式: 陆旭, 于斌, 段振华, 王德奎, 陈矗, 崔进. 智能规划中面向简单偏好的高效求解方法. 软件学报, 2023, 34(7): 3099–3115. http://www.jos.org.cn/1000-9825/6859.htm

英文引用格式: Lu X, Yu B, Duan ZH, Wang DK, Chen C, Cui J. Efficient Approach for Solving Simple Preference in AI Planning. Ruan Jian Xue Bao/Journal of Software, 2023, 34(7): 3099−3115 (in Chinese). http://www.jos.org.cn/1000-9825/6859.htm

Efficient Approach for Solving Simple Preference in AI Planning

LU Xu[1,2], YU Bin[1,2], DUAN Zhen-Hua[1,2], WANG De-Kui[3], CHEN Chu[4], CUI Jin[5]

[1](School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

[2](State Key Laboratory of Integrated Service Networks (Xidian University), Xi'an 710071, China)

[3](School of Information Science and Technology, Northwest University, Xi'an 710127, China)

* 基金项目: 国家自然科学基金(61806158, 61732013, 62172322, 62002290); 中国博士后科学基金(2019T120881, 2018M643585); 国家重点研发计划(2018AAA0103202); 陕西省重点科技创新团队项目(2019TD-001); 陕西省自然科学基础研究计划(2021JQ-208); 山东省自然科学基金(ZR2020MF030, ZR2018PF007); 赛尔网络下一代互联网技术创新项目(NGII20190407); 陕西省教育厅专项科研计划(21JK0844)

本文由"形式化方法与应用"专题特约编辑董云卫教授、刘关俊教授、毛晓光教授推荐.

收稿时间: 2022-09-04; 修改时间: 2022-10-08; 采用时间: 2022-12-05; jos 在线出版时间: 2022-12-30

[4](School of Computer Science, Qufu Normal University, Qufu 273165, China)
[5](School of Computer Science, Xi'an Shiyou University, Xi'an 710065, China)

Abstract: AI planning, or planning for short, is an important branch of AI and widely applied in many fields, e.g., job shop scheduling, transportation scheduling, robot motion planning, aerospace mission planning, etc. A plan (a sequence of actions) must achieve all goals eventually in traditional planning, where such goals are called hard goals. Nevertheless, in many practical problems, the key focus is not only on the realization of goals as soon as possible and the reduction of the cost of plans as low as possible, but also on other factors, e.g., resource consumption or time constraint. To this end, the concept of simple preference which is also called soft goals is introduced. In contrast to hard goals, a simple preference is allowed to be violated by a plan. In essence, simple preferences are used to measure the quality of plans, without affecting the existence of plans. Current research on simple preferences makes less progress and the quality of plans are often unsatisfactory. This study proposes an efficient approach for solving simple preferences which are modeled as a part of classical planning models. Moreover, SMT (satisfiability modulo theories) solver is employed to recognize the mutual exclusion relations among simple preferences for the purpose of preference reduction, relieving the burden of planers. The major advantages of this approach lie in: on one hand, the state space is largely reduced due to the pre-tailoring of simple preferences, and on the other hand, the existing fast planners can be utilized and there is no need to design specialized planning algorithm. The experimental results on benchmarks show that the proposed approach has sound performance in improving the quality of plans, especially suited for the situation where simple preferences are not independent of each other.

Key words: AI planning; simple preference; SMT

智能规划(AI planning)[1], 简称规划, 是人工智能领域的一个重要研究分支. 一般来说, 规划问题模型由初始状态、目标集和动作集组成. 规划过程的基本思想是: 根据期望实现的目标集, 在给定约束条件的限制下, 综合制定出实现目标集的动作序列, 即规划解. 规划在各领域均有广泛应用, 如工厂车间作业调度、物资运输调度、机器人动作规划以及航空航天任务规划等.

规划的目标主要有硬目标(hard goal)和软目标(soft goal)两种类型: 前者为必须实现的目标, 而后者则是可选择实现的目标. 例如, 某机器人的硬目标是制造一些产品, 其软目标是制造产品后保持工作台的清洁整齐. 软目标与硬目标最重要的不同之处在于: 即便违反软目标, 也不会造成规划解失效. 换句话说, 软目标用于衡量规划解质量的优劣, 而不会影响规划解是否存在. 前述机器人例子的规划解必须完成产品制造的任务, 同时保持工作台清洁整齐为最优, 不满足清洁整齐规划解仍然成立, 但是会认为该规划解而质量稍差. 规划领域公认的标准建模语言 PDDL (planning domain definition language)在其 3.0 版本[2]中引入了"偏好"的概念, 采用 Preference 关键字定义, 其中的简单偏好实际上就是带惩罚权重的软目标. 若最终没有实现偏好, 则在规划解的开销中增加相应权重, 从而降低规划解的质量. 任意偏好都不被强制要求实现, 但是期望通过规划实现最优或较优(惩罚权重之和最小或较小)的偏好组合. 国际规划竞赛 IPC (Int'l planning competition)曾专门举办了针对偏好的竞赛, 使得偏好的研究得到了持续关注.

显然, 偏好的引入改变了衡量规划解质量的传统标准. 在偏好存在的情况下, 实现硬目标动作序列开销最低的解并不一定是最优解, 原因是并未考虑偏好是否实现. 此时的最优解应该在动作序列开销和偏好惩罚权重之间取得权衡. 这相当于一个约束优化问题, 试图在尽量满足更多偏好的同时保持较低的动作序列开销. 本文主要研究求解简单偏好的高效规划方法, 其主要思想是: 首先对偏好集进行约简; 然后将约简后的偏好组合编码为经典规划(classical planning)模型中, 并利用现有规划器进行求解.

本文第 1 节简要介绍相关研究工作. 第 2 节回顾与规划及简单偏好相关的基本概念. 第 3 节和第 4 节分别详细阐述简单偏好集的约简方法及简单偏好的经典规划模型编码方法. 第 5 节通过基准实验说明本文所提方法的可行性和有效性. 最后总结全文, 并对未来可能的研究方向进行初步探讨.

1 相关研究工作

偏好机制是 PDDL3.0 语言新增的两大特性之一, 主要以轨迹约束的形式定义, 要求从初始状态出发到最终状态的状态序列需满足一定限制, 例如 PDDL 3.0 定义的偏好有(at-end ϕ)、(always ϕ)、(sometime ϕ)、(at-most-once ϕ)、(sometime-before $\phi\psi$)、(sometime-after $\phi\psi$)等, 其中, ϕ和ψ为谓词逻辑公式. 这些偏好的语义比较容易

理解, 顾名思义, 如(at-end ϕ)表示最终ϕ成立, (at-most-once ϕ)表示ϕ最多只能在一个时间段内保持成立, (sometime-before $\phi\psi$)表示ψ必须在ϕ之前成立. 偏好(at-end ϕ)也称简单偏好或软目标, 本文针对简单偏好进行研究. 为了避免混淆, 后续文中统一采用简单偏好指代软目标.

Baier 等人从规划语言和规划算法这两个角度概述了偏好规划的相关工作, 并基于定量和定性语言提出了不同的偏好模型[3]: 在定量语言中, 采用数值表示偏好实现的程度, 如马尔可夫决策过程的奖励; 在定性语言中, 偏好实现的程度通过性质的可满足性表示, 如 PDDL 3.0 的偏好可以采用线性时序逻辑 LTL 的子集描述. Son 等人提出了一种声明式语言 PP 定义偏好[4], 允许表达多维偏好以及偏好优先级, 这是其他偏好模型所不具备的. *PP* 偏好的求解采用逻辑编程框架回答集编程(answer set programming)实现. 这种方法的缺点是可扩展性不强, 其模型并不是标准的 PDDL 模型. Baier 等人扩展了 TLPlan 规划器, 提出了一些偏好专用的启发式算法, 实现的规划器称作 HPlan-P, 在第 5 届 IPC 取得了不错的成绩[5]. 文献[6]通过扩展 PDDL3.0, 使其支持 HTN (hierarchical task networks)描述的偏好形式, 并提出了一种分支限界算法以及一系列启发式搜索机制. 该方法在工具层面扩展了 SHOP2 规划器, 形成了 HTNPlan-P 规划器. 基于 HTN 框架的规划方法本质上属于知识型规划方法, 在规划前需要提供一定的引导知识, 因此, 其性能相较通用的偏好规划器具有一定的优势.

Wright 等人提出了一种基于时序逻辑的偏好求解方法, 其基本思路是: 将偏好描述为时序逻辑公式, 公式又可以转换为有限自动机, 最后将自动机的节点和迁移编码为经典规划模型[7]. 本文借鉴了这种编码方法, 提出的方法复杂度较低, 不存在自动机转换的步骤. Percassi 等人也提出了一种偏好编码方法[8], 与文献[7]的不同之处在于: 该方法没有将偏好转换为自动机, 而是直接针对 PDDL3.0 定义的特定偏好提出专用的编码算法, 将偏好编码为规划动作中额外的动作效果. 此方法在求解效率方面高于文献[7], 但在偏好的表达能力方面有所欠缺. 文献[9]针对 PDDL3.0 的偏好提出了求解算法, 利用辅助定义的谓词记录约束满足的情况. 然而, 该算法变相地要求偏好不能违反, 实际上是一种硬目标求解的模式. 求解偏好面临的最大问题是: 动作开销是状态依赖的, 而不像传统规划中定义为固定开销. 为了解决此问题, 研究人员定义了新的启发式函数和代价法(additive heuristic)的推广, 结合 EVMDD (edge-valued multi-valued decision diagrams)技术, 更为高效地求解偏好[10]. 目前, 还存在许多规划的研究工作基于 SMT (satisfiability modulo theories)技术, 但是聚焦在经典硬目标求解[11]或具体规划应用[12], 未检索到针对偏好提出的 SMT 方法.

2 研究背景

2.1 谓词逻辑

谓词逻辑在命题逻辑的基础上扩充了量词和项, 可以描述更为丰富的推理形式. 令 V 为变量集合, C 为常量集合, 谓词逻辑的语法定义如下:

$$e::=c|x\ \phi::=p(e_1,\ldots,e_n)|\neg\phi|\phi_1\wedge\phi_2|\exists x{:}\phi,$$

其中, $c\in C$ 为个体常量, $x\in V$ 为个体变量, c 和 x 构成了项 e, $p(e_1,\ldots,e_n)$为谓词, p 是谓词的名称, $e_1,\ldots,e_n$ 是谓词的项($e_1,\ldots,e_n$ 在谓词中出现的顺序不做限制). True、false 和其他逻辑操作符“$\vee$” “$\rightarrow$” “$\leftrightarrow$” “$\forall$”可以由ϕ中定义的基本操作符导出. 例如, $\phi_1\vee\phi_2\equiv\neg(\phi_1\wedge\phi_2)$, $\phi_1\rightarrow\phi_2\equiv\neg\phi_1\vee\phi_2$, $\phi_1\leftrightarrow\phi_2\equiv(\phi_1\rightarrow\phi_2)\wedge(\phi_2\rightarrow\phi_1)$, $\exists x{:}\phi\equiv\forall x{:}\neg\phi$. 表示具体性质与关系的谓词称为谓词常项, 其包含的项均为个体常量; 反之, 包含项均为个体变量的谓词称为谓词变项. 符号 $Pred(\phi)$表示ϕ中出现的谓词名称, 符号 $\vec{e}$ 作为 $e_1,\ldots,e_n$ 的缩写.

令$\sigma{:}V\rightarrow C$ 为变量到常量的映射, 表示变量的取值. 给定谓词常项集合 P_{const} 与σ, 谓词逻辑的可满足性语义定义如下.

- $(P_{const},\sigma)[c]=c$ $(P_{const},\sigma)[x]=\sigma(x)$.
- $(P_{const},\sigma)\models p(e_1,\ldots,e_n)$ iff $p((P_{const},\sigma)[e_1],\ldots,(P_{const},\sigma)[e_n])\in P_{const}$.
- $(P_{const},\sigma)\models\neg\phi$ iff not $(P_{const},\sigma)\models\phi$.
- $(P_{const},\sigma)\models\phi_1\wedge\phi_2$ iff $(P_{const},\sigma)\models\phi_1$ and $(P_{const},\sigma)\models\phi_2$.
- $(P_{const},\sigma)\models\exists x{:}\phi$ iff there exists $c\in C$ such that $(P_{const},\sigma)\models\phi(c/x)$.

虽然谓词逻辑的可满足性问题是不可判定的, 但是如果将个体域限制为有限, 则该问题变为可判定的. 本文采用谓词逻辑描述规划问题中的简单偏好, 归因于规划问题模型的有限性, 简单偏好公式的可满足性是可判定的. 此外, 简单偏好公式中不允许出现自由变量, 因此$\sigma=\varnothing$. $(P_{const},\varnothing)\models\phi$可简写为 $P_{const}\models\phi$.

2.2 规划问题

本节将介绍经典规划问题以及简单偏好的形式化定义, 作为后续章节的理论基础.

定义 1(经典规划). 一个经典规划问题(classical planning problem) $\mathcal{P}$是一个四元组(P,A,I,HG), 其中, P 是有限的事实(fluent)集合,通常采用谓词常项表示; A 是有限的动作集合; $I\subseteq P$ 是初始状态; $HG\subseteq P$ 是硬目标集合. P 和 A 被称为问题域(domain), 共享相同问题域的不同规划问题仅在 I 和 HG 上有所区别, I 和 HG 被称为问题实例(instance). P 中所有状态可枚举为 $S=2^P$, 注意, 每个状态 $s\in S$ 仅需指定成立的谓词, 符合封闭世界假设(closed world assumption), 即凡是未出现在 s 中的谓词均不成立. 给定状态 $s\subseteq P$ 和事实 $p(\vec{c})\in P$, s 满足 $p(\vec{c})$ 当且仅当 $s\models p(\vec{c})$, s 满足 $\neg p(\vec{c})$ 当且仅当 $s\models\neg p(\vec{c})$. 令 P 的字集合表示为 $Lits(P)=P\cup\{\neg p(\vec{c})\mid p(\vec{c})\in P\}$, s 满足字集合 $L\subseteq Lits(P)$当且仅当对于任意 $l(\vec{c})\in L$, $s\models l(\vec{c})$. 为了方便叙述, 本文直接采用规划问题指代经典规划问题.

对于每个动作 $a\in A$, $a=(pre(a),eff(a),cost(a))$. $pre(a)\subseteq Lists(P)$被称为前置条件, 意味着只有当前状态满足 $pre(a)$时, a 才有机会执行. $eff(a)$是动作效果, 由一系列条件效果组成, 定义为 $eff(a)=\{cond_i\triangleright l_i(\vec{c}_i)\mid i=1,...,n\}$, 其中, $cond_i$ 是谓词逻辑公式, $l_i(\vec{c}_i)\in Lits(P)$, $n\in\mathbb{N}$为条件效果的数量. a 执行结束后, 若 $cond_i$ 成立, 则 $l_i(\vec{c}_i)$ 置为真, 并忽略 $l_i(\vec{c}_i)$ 在上一状态的真值. 若 $cond_i$=true, $cond_i\triangleright l_i(\vec{c}_i)$ 可缩写为 $l_i(\vec{c}_i)$. $cost(a)\in\mathbb{N}$是与 a 相关联的动作权值, 其含义是赋予 a 执行的开销. 需要注意的是, 经典规划中仅支持动作权值是固定值.

如果状态 $s\in S$ 满足 $pre(a)$, 则称动作 a 在 s 下是可执行的. 令 s'为 a 在 s 下执行的结果, 定义为 $s'=s\setminus\{p(\vec{c})\mid(cond\triangleright\neg p(\vec{c}))\in eff(a),s\models cond\}\cup\{p(\vec{c})\mid(cond\triangleright p(\vec{c}))\in eff(a),s\models cond\}$, 采用 $app(s,a)=s'$表示. 假设 $\pi=\langle a_1,\ldots,a_n\rangle$为动作序列. π的开销是π中所有动作的权值之和, 即 $cost(\pi)=\sum_{i=1}^{n}cost(a_i)$. 若从 I出发, 经过执行 π到达的状态满足 HG, 即 $app(\ldots app(I,a_1)\ldots,a_n)\models HG$, 则称$\pi$是 P 的规划解. 为简单起见, 从任意状态 s 出发, 执行某动作序列π到达的状态记为 $app(s,\pi)$.

定义 2(简单偏好). 给定一个规划问题$\mathcal{P}=(P,A,I,HG)$, 可以为其扩展定义简单偏好集合 SP, 其中的所有元素为谓词逻辑公式, 且公式中的变量均为约束变量. 这类问题被称为带有简单偏好的规划问题. 每个简单偏好 $sp\in SP$ 都会关联一个惩罚权值 $penalty(sp)\in\mathbb{N}$. 假设π是 P 的一个规划解, π的惩罚定义为 $penalty(\pi)=\sum_{sp\in SP\text{且not }app(I,\pi)\models sp}penalty(sp)$. 规划解的质量能够采用开销和惩罚共同衡量, 即 $quality(\pi)=cost(\pi)+penalty(\pi)$. 如果不存在比 $quality(\pi)$更小的其他规划解, 则称π为最优解.

提高规划解质量的途径就是通过降低动作开销和简单偏好惩罚这两方面完成, 而为了满足简单偏好可能会执行额外的动作(完成硬目标的动作除外), 增加动作开销, 因此需要在二者之间取得平衡. 如不做特殊说明, 本文后续采用偏好指代简单偏好.

例 1(规划问题举例): 采购员问题(traveling and purchase problem, TPP)是经典旅行商问题的一个变种. 如图 1 所示, 已知不同种类的商品集合、不同仓库的集合以及不同市场组成的集合, 其中, 每个市场能够提供有限数量、有限种类的商品. TPP 问题的目标是: 购买指定数量、指定种类的商品, 并尽可能地最小化路由代价和购买代价. 表 1 给出一个带有偏好的 TPP 问题的具体定义. 这里借鉴 PDDL 中谓词和动作参数化的表达方式,事实集合中的谓词均为谓词变项, 动作集合中的动作也通过变量作为参数简化定义. 实际上, 事实集合与动作集合需继续通过参数的实例化得到. 所有动作的开销相同且为 1, 前置条件列表的谓词是合取关系. 硬目标规定了商品 $goods_3$ 存储在仓库中的数量, 而偏好对商品 $goods_1$ 和 $goods_2$ 存储在仓库中的数量以及二者之间的数量关系做出了约定, 同时也定义了相应的惩罚. 容易求得, 此问题的一个规划解如下, 且为最优解,其质量为 $quality(\pi)=cost(\pi)+penalty(\pi)=11+4=15$:

$\pi=\langle drive(truck_1,depot_1,market_1),buy(truck_1,goods_3,market_1,lelve_1,level_2,level_0,level_1),$
$load(goods_3,truck_1,market_1,lelve_0,level_1,level_0,level_1),buy(truck_1,goods_2,market_1,lelve_1,level_2,level_0,level_1),$
$load(goods_2,truck_1,market_1,lelve_0,level_1,level_0,level_1),buy(truck_1,goods_1,market_1,lelve_0,level_1,level_0,level_1),$
$load(goods_1,truck_1,market_1,lelve_0,level_1,level_0,level_1),drive(truck_1,market_1,depot_1),$
$unload(goods_1,truck_1,depot_1,lelve_0,level_1,level_0,level_1),unload(goods_2,truck_1,depot_1,lelve_0,level_1,level_0,level_1),$
$unload(goods_3,truck_1,depot_1,lelve_0,level_1,level_0,level_1)\rangle$.

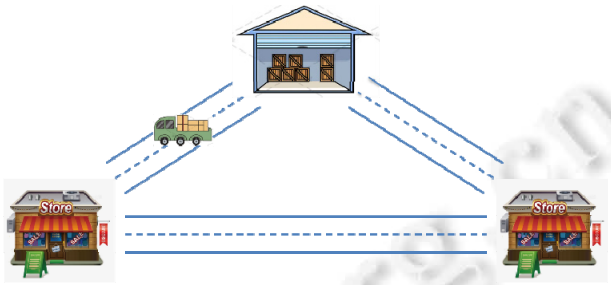

图 1 采购员问题图示

表 1 TPP 问题实例定义

	形式化定义	含义
事实集合	$loaded(g,t,l)$	卡车 t 装载商品 g 的数量为 l
	$ready\_to\_load(g,m,l)$	市场 m 售出商品 g 的数量为 l (准备由卡车装载运往别处)
	$stored(g,l)$	仓库中存储商品 g 的数量为 l
	$on\_sale(g,m,l)$	商店 m 在售商品 g 的数量为 l
	$next(l_1,l_2)$	数量 l_1 等于数量 l_2 加 1(经典规划问题可通过二元谓词模拟连续整数之间的关系)
	$at(t,p)$	卡车 t 在地点 p 处
	$connected(p_1,p_2)$	从地点 p_1 到地点 p_2 存在一条路径
动作集合	$drive(t,from,to)$ $pre(drive)=\{at(t,from),connected(from,to)\}$ $eff(drive)=\{\neg at(t,from),at(t,to)\}$ $cost(drive)=1$	卡车 t 从地点 $from$ 行驶到地点 to
	$load(g,t,m,l_1,l_2,l_3,l_4)$ $pre(load)=\{at(t,m),loaded(g,t,l_3),$ $ready\text{-}to\text{-}load(g,m,l_2),next(l_2,l_1),next(l_4,l_3)\}$ $eff(load)=\{\neg loaded(g,t,l_3),loaded(g,t,l_4),$ $\neg ready\text{-}to\text{-}load(g,m,l_2),ready\text{-}to\text{-}load(g,m,l_1)\}$ $cost(load)=1$	卡车 t 在市场 m 装载商品 g, 使得 g 准备装载的数量由 l_2 减少为 l_1, t 装载 g 的数量由 l_3 增加为 l_4
	$unload(g,t,d,l_1,l_2,l_3,l_4)$ $pre(unload)=\{at(t,d),loaded(g,t,l_2),$ $stored(g,l_3),next(l_2,l_1),next(l_4,l_3)\}$ $eff(unload)=\{\neg loaded(g,t,l_2),loaded(g,t,l_1),$ $\neg stored(g,l_3),stored(g,l_4)\}$ $cost(unload)=1$	卡车 t 将商品 g 卸载在仓库 d, 使得 t 装载 g 的数量由 l_1 减少为 l_2, 完成存储的 g 的数量由 l_3 增加为 l_4
	$buy(t,g,m,l_1,l_2,l_3,l_4)$ $pre(buy)=\{at(t,m),on\text{-}sale(g,m,l_2),$ $ready\text{-}to\text{-}load(g,m,l_3),next(l_2,l_1),next(l_4,l_3)\}$ $eff(buy)=\{\neg on\text{-}sale(g,m,l_2),on\text{-}sale(g,m,l_1),$ $\neg ready\text{-}to\text{-}load(g,m,l_3),ready\text{-}to\text{-}load(g,m,l_4)\}$ $cost(buy)=1$	在市场 m 购买商品 g, 使得 g 准备装载的数量由 l_3 增加为 l_4, 市场在售 g 的数量由 l_2 减少为 l_1

表 1 TPP 问题实例定义(续)

	形式化定义
初始状态	$\{next(l_1,l_0),next(l_2,l_1),$ $ready\text{-}to\text{-}load(goods_1,market_1,level_0),ready\text{-}to\text{-}load(goods_1,depot_1,level_0),$ $ready\text{-}to\text{-}load(goods_2,market_1,level_0),ready\text{-}to\text{-}load(goods_2,depot_1,level_0),$ $ready\text{-}to\text{-}load(goods_3,market_1,level_0),ready\text{-}to\text{-}load(goods_3,depot_1,level_0),$ $stored(goods_1,level_0),stored(goods_2,level_0),stored(goods_3,level_0),$ $loaded(goods_1,truck_1,level_0),loaded(goods_2,truck_1,level_0),loaded(goods_3,truck_1,level_0),$ $connnected(depot_1,market_1),connnected(market_1,depot_1),$ $on\text{-}sale(goods_1,market_1,level_0),\ on\text{-}sale(goods_2,market_1,level_0),on\text{-}sale(goods_3,market_1,level_0),$ $at(truck_1,depot_1)\}$
硬目标集合	$\{stored(goods_3,level_1)\}$
偏好集合	$\{sp_1=stored(goods_1,level_1),sp_2=stored(goods_2,level_2),$ $sp_3=stored(goods_2,level_1),sp_4=stored(goods_2,level_2),$ $sp_5=stored(goods_1,level_1)\rightarrow stored(goods_2,level_1),sp_6=stored(goods_1,level_2)\rightarrow stored(goods_2,level_2)\}$ $penalty(sp_1)=penalty(sp_3)=1,\ penalty(sp_2)=penalty(sp_4)=2,\ penalty(sp_5)=penalty(sp_6)=4$

3 基于 SMT 的简单偏好约简方法

一般来说, 硬目标之间的独立性相对较强, 每个硬目标的实现不会使得其他目标无法实现, 否则, 规划问题无解. 然而, 由于规划问题对偏好是否最终实现不做强制要求, 因此偏好的设置允许出现潜在的相互影响. 例如, 例 1 中的 sp_1 和 sp_2 不可能同时实现, 因为同一时刻 $goods_1$ 存储的数量是唯一的, 二者之间是互斥的关系. 又如: 例 1 中的 sp_5 约定了 $goods_1$ 存储数量和 $goods_2$ 存储数量的关系, 从而影响 sp_1, sp_2, sp_3, sp_4 的实现, 它们之间是关联的关系. 偏好越多, 意味着求解的目标越多, 规划器的效率将会大概率随之降低. 如果提前约简偏好集合, 从而减少惩罚值, 能够对提高规划效率和规划解质量起到积极作用. 本节将会详细阐述如何利用 SMT 技术约简偏好集合.

3.1 SMT求解技术

SMT, 即可满足性模理论, 是指在相关背景理论(如算术、位向量、数组和未解释函数等)下, 判定一阶逻辑公式的可满足性问题, 属于约束满足问题的一种形式. SMT 采用一阶逻辑公式描述问题约束, 在优化问题求解[13,14]、程序验证[15,16]、静态分析[17,18]等领域有突出优势, 这些领域的实际问题都可以建模为约束可满足问题.

实现 SMT 判定算法的工具被称为 SMT 求解器, 不仅可以检查一阶逻辑公式的可满足性, 还能在公式可满足时给出一组满足约束的随机模型. 目前, 主流的 SMT 求解器有 Z3[19]、CVC4[20]、Yices2[21]、MathSAT4[22]、Boolector[23]等. 随着 SMT 背景理论的逐渐成熟和判定算法的不断发展, 这些求解器已经具备处理大规模工业化的实际应用问题的能力. SMT 求解器也被集成到一些知名工具中, 例如微软开发的 PEX 工具(自动单元测试)[24]和慕尼黑大学开发的 CPAchecker(软件模型检测)[25]. 本文第 2 节定义的谓词逻辑在 SMT 求解能力的范畴之内.

3.2 偏好关系定义

给定规划问题定义的偏好集合, 约简算法的核心思想是: 结合 SMT 求解器分析不同偏好之间的关系, 据此尝试求解获得惩罚较小的偏好组合, 从而达到缩减偏好数量的效果. 仅仅将偏好作为逻辑公式进行可满足性判定得不到足够的偏好关系信息, 还需要在规划问题蕴含的约束辅助下进行判定. 因此, 下面首先给出由规划问题定义提取的规划约束, 称为规划公理.

定义 3(规划公理). 给定带偏好的规划问题$\mathcal{P}$=(P,A,I,HG,SP). 静态公理 $static(P)$是指在规划过程中真值保持不变的谓词常项集合, 即 $static(P)=\{p(\vec{c})\mid \text{for all } a\in A \text{ and } ce\in eff(a): ce=cond \rhd l(\vec{c}_1) \text{ and } p(\vec{c}_1)\neq l(\vec{c}_1) \text{ and } p(\vec{c})\neq\neg l(\vec{c}_1)\}$ (根据逻辑语义, $\neg\neg p=p$). 反之, 除静态公理之外的谓词称为动态公理 $dynamic(P)=P\backslash static(P)$.

此外, 还存在一种特殊公理, 用于表示一组同名谓词在每个状态下仅有一个谓词成立. 给定谓词名 p, 若下式恒成立, 则称该式为 p 的唯一性公理. $\mathcal{P}$中所有唯一性公理记为 $unique(P)$:

$$\forall x_1,...,x_{n1},y_1,...,y_{n2}: p(x_1,...,x_{n1},y_1,...,y_{n2}) \rightarrow \neg\exists y_1',...,y_{n2}': y_1 \neq y_1' \wedge ... \wedge y_{n2} \neq y_{n2}' \wedge p(x_1,...,x_{n1},y_1',...,y_{n2}').$$

显然, P=$static(P)\cup dynamic(P)$, 且上述 3 种规划公理均可通过对规划模型进行语法检查得到. 例如, 给定谓词名称 p, 唯一性公理存在的条件是: 对于任意动作的任意条件效果, 如果存在将 p 结合某些常项产生的谓词置为真, 那么将 p 结合其他常项产生的谓词置为假.

例 2(规划公理举例): 例 1 中的谓词变项 $next(l_1,l_0)$和 $connect(p_1,p_2)$实例化后的谓词常项是静态公理, 所有动作不会改变它们的真值. $at(t,p)$实例化后的谓词常项是动态公理, 因为动作 $drive$ 改变了其真值. $stored(g,l)$对应一个唯一性公理$\forall g,l:stored(g,l)\rightarrow\neg\exists l':l\neq l'\wedge stored(g,l')$, 因为对于任意 g, 同一时刻仅存在一个 l 使之成立. 动作 $unload$将 $stored(g,l)$置为真, 并将 $stored(g,l')$置为假, 二者不会同时为真. 同理, $loaded(g,t,l)$, $at(t,p)$, $ready\_to\_load(g,m,l)$, $on_sale(g,m,l)$均存在对应的唯一性公理.

定义 4(偏好的互斥关系). 给定一个带偏好的规划问题$\mathcal{P}$=(P,A,I,HG,SP), 若其中两个偏好 $sp_1,sp_2\in SP$ 不能同时被实现, 则称二者是互斥关系.

总的来讲, 识别出所有的互斥偏好难度较大, 但是可通过求解偏好的可满足性提前发现部分互斥偏好对. 下面给出判断偏好互斥的规则: 给定一个带偏好的规划问题$\mathcal{P}$=(P,A,I,HG,SP)及 $sp_1,sp_2\in SP$, 如果 $static(P)\wedge unique(P)\wedge sp_1\wedge sp_2$ 是不可满足的, 那么 sp_1 和 sp_2 一定为互斥关系.

定义 5(偏好的关联关系). 给定一个带偏好的规划问题$\mathcal{P}$=(P,A,I,HG,SP), 若其中偏好 $sp_1,\ldots,sp_n\in SP$ 中含有共享谓词名, 即 $Pred(sp_1)\cap\ldots\cap Pred(sp_n)\neq\varnothing$, 则称它们是关联关系.

例 3(偏好关系举例): 根据定义 4 和定义 5, 例 1 中的各偏好关系列举如下.

(1) 互斥关系: sp_1 和 sp_2, sp_3 和 sp_4.

(2) 关联关系: sp_1, sp_2, sp_3, sp_4, sp_5, sp_6.

3.3 偏好约简算法设计

在偏好关系的基础上, 本文提出了偏好集合的约简方法, 该方法的核心思想概括如下(如图 2 所示).

(1) 偏好初步约简: 从偏好集合中去除两类偏好: 一类是自身不可满足的偏好, 另一类是无法实现的偏好. 采用 SMT 求解器单独求解每个偏好公式, 删除不可满足的偏好. 将每个偏好公式定义为派生谓词, 并将其作为硬目标, 交由规划器单独求解, 删除无法实现的偏好.

(2) 偏好公理识别: 从规划模型中获得静态公理和唯一性公理. 更具体地说, 静态公理从事实集合和动作集合中获得, 唯一性公理从动态公理和动作集合中获得. 为了进一步简化偏好公理, 公理中仅保留偏好集合中出现的谓词.

(3) 偏好划分: 识别偏好之间的互斥关系, 并据此对偏好集合进行划分, 使得属于同一子集合的偏好之间均为互斥关系.

(4) 偏好组合: 从划分后的偏好集合中选取偏好进行组合, 选取的标准是惩罚值高且经 SMT 求解器求解(具有关联关系的偏好组)是可满足的.

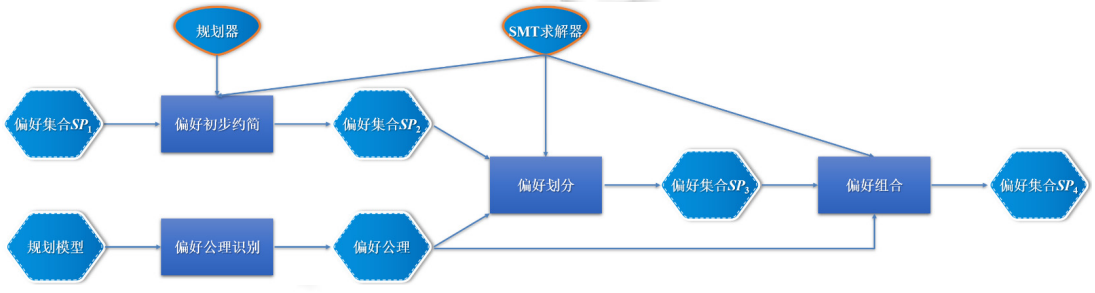

图 2 偏好集合约简方法框架

步骤(1)和步骤(2)相对简单, 这里不做赘述. 本节将重点阐述步骤(3)和步骤(4). 算法 1 和算法 2 分别展示了偏好划分和偏好组合的过程.

算法 1. 互斥偏好集合获取 *GetMutexSP*.

输入: 带偏好的规划问题$\mathcal{P}$=(P,A,I,HG,SP), 其中, SP={sp_1,…,sp_n}.

输出: 互斥偏好集合 SP_{mutex}={SP_1,…,SP_m}, 对于任意 i∈{1,…,m}, SP_i⊆SP, 且$\bigcup_{i=1}^{m} SP_i = SP$.

1 SP_{mutex}={{sp_1},…,{sp_n}}; $oldSP_{mutex}$:=SP_{mutex};
2 **while** (true) /*寻找不动点, 直至 SP_{mutex} 不再变化*/
3 $copySP_{mutex}$:=SP_{mutex};
4 **while** ($copySP_{mutex}$≠∅) /*合并 SP_{mutex} 中存在的互斥集合*/
5 $subSP'$∈$copySP_{mutex}$;
6 $copySP_{mutex}$:=$copySP_{mutex}$\$subSP'$;
7 **for** $subSP$ in SP_{mutex}
8 **if** ($subSP'$=$subSP$) /*偏好集合自身不满足互斥*/
9 continue;
10 **end if**
11 **if** ($static$(P)∧$unique$(P)∧sp'∧sp is *unsat* for each sp'∈$subSP'$ and sp'∈$subSP$) **then**
/*两个偏好集合满足两两元素互斥*/
12 $combSP$=sp∪sp';
13 SP_{mutex}:=SP_{mutex}∪$combSP$;
14 SP_{mutex}:=SP_{mutex}\$sp$\$sp'$;
15 break;
16 **end if**
17 **end for**
18 **end while**
19 **if** (SP_{mutex}=$oldSP_{mutex}$) **then** /*判断是否到达不动点*/
20 break;
21 **end if**
22 $oldSP_{mutex}$:=SP_{mutex};
23 **end while**
24 return SP_{mutex};

算法 2. 偏好组合选择 *SelectSP*.

输入: 带偏好的规划问题$\mathcal{P}$=(P,A,I,HG,SP), 互斥偏好集合 SP_{mutex}={SP_1,…,SP_m}, 期望输出的偏好组合集合数量 k.

输出: 偏好组合集合 SP_{opt} = {SP_{opt_1},...,SP_{opt_k}}, 对于任意 i∈{1,…,k}, SP_{opt_i} ⊆ SP

1 n:=0; /*偏好组合集合计数*/
2 $penaltySP_{mutex}$={$penalty$(SP_1),…,$penalty$(SP_m)};
3 $penaltySP_i$:= {$penalty(sp_1^i)$,..., $penalty(sp_{|SP_i|}^i) \mid sp_j^i \in SP_i, 1 \leqslant j \leqslant |SP_i|, penalty(sp_1^i)$ is ranked in descending order;
4 $bound := \sum_{i=1}^{m} penalty(sp_1^i)$; /*设置初始的最大惩罚值*/
5 $sumPenalty$[m+1][$bound$+1]:={{0}}; /*约简过程中偏好的惩罚值记录*/
6 $SP_{opt} := \{SP_{opt_1} := \varnothing, ..., SP_{opt_k} := \varnothing\}$;

7 **while** ($n<k$)
8 **for** (i:=1; $i\leqslant m$; i++)
9 **for** (j:=1; $j\leqslant bound$; j++)
10 **if** ($\forall penalty(sp)\in penalty(SP_i)$:$j<penalty(sp)$) **then**
11 $sumPenalty[i][j]=sumPenalty[i][j-1]$;
12 **else**
13 $sumPenalty[i][j]:=\max(sumPenalty[i-1][j], j \geqslant penalty(sp_1^i)?$
$sumPenalty[i-1][j-penalty(sp_1^i)]+penalty(sp_1^i):0,..., j \geqslant penalty(sp_{|SP_i|}^i)?$
$sumPenalty[i-1][j-penalty(sp_{|SP_i|}^i)]+penalty(sp_{|SP_i|}^i):0)$;
14 **end if**
15 **end for**
16 **end for**
17 **for** (i:=m, j:=$bound$; $i\geqslant 0$; i--)
18 **if** ($sumPenalty[i][j]=sumPenalty[i-1][j]$) **then**
19 continue;
20 **else if** (for some r : $j-penalty(sp_r^i)\geqslant 0$ && $sumPenalty[i][j]=sumPenalty[i-1][j-penalty(sp_r^i)]+penalty(sp_r^i)$) **then**
21 $SP_{opt_{n+1}}=SP_{opt_{n+1}}\cup\{sp_r^i\}$;
22 $j:=j-penalty(sp_r^i)$;
23 **end if**
24 **end for**
25 **if** ($\wedge_{sp\in SP_{opt_{n+1}}} sp\wedge static(P)\wedge unique(P)$ is sat) **then** /*求解具有关联关系的偏好组合*/
26 n++;
27 $SP_{opt}=SP_{opt}\cup SP_{opt_{n+1}}$;
28 **end if**
29 $bound$--;
30 **end while**
31 return SP_{opt};

算法 1 的输入为带偏好的规划问题$\mathcal{P}$=(P,A,I,HG,SP), 通过 SMT 求解器识别不同偏好的互斥关系, 输出互斥偏好集合 $SP_{mutex}=\{SP_1,\ldots,SP_m\}$, 保证 SP_{mutex} 是原偏好集合 SP 的划分, 且 SP_{mutex} 的各元素中所有偏好互斥. 算法的框架主要包含两层循环: 外层循环和内层循环. 第 1 行初始化 SP_{mutex}, 并采用另一个集合 $oldSP_{mutex}$ 记录上一次外层循环迭代时 SP_{mutex} 的划分情况, 目的是判断当前是否到达不动点. 第 2–23 行为外层循环,当 SP_{mutex} 不再变化时(找到不动点)跳出循环(第 19–21 行), 算法返回; 否则, 对 $oldSP_{mutex}$ 赋值为 SP_{mutex}(第 22 行), 进行下一次迭代. 第 4–18 行为内层循环, 目的是逐步对 SP_{mutex} 进行划分, 合并 SP_{mutex} 中存在的互斥偏好集合. 在内层循环开始之前(第 3 行), 定义新集合 $copySP_{mutex}$ 并赋值为 SP_{mutex}, 用于在内层循环中与 SP_{mutex} 元素的比对. 每次内层迭代取出 $copySP_{mutex}$ 中的一个元素(第 5–6 行), 依次与 SP_{mutex} 中的所有元素比较(第 7–17 行), 直至 $copySP_{mutex}$ 为空. 若 $copySP_{mutex}$ 中取出的元素与 SP_{mutex} 中的某元素满足互斥(第 11 行), 则合并两元素(第 12–14 行), 接着进行下一次内层循环迭代.

算法 1 输出的互斥偏好集合 SP_{mutex} 作为算法 2 的输入. 通过比较不同偏好组合的惩罚值, 算法 2 输出前 k 个最优的偏好组合, 其中, k 为全局配置参数, 由算法使用者设定. 算法 2 仍然从 SMT 求解的角度定义最优偏好组合, 不等价于最优规划解. 若考虑效率问题, k 值设置偏小为佳; 若考虑最优求解覆盖问题, 则可将 k 值增

大. 前 6 行的任务是算法的初始化, 包括: 计数变量 n 的初始化(第 1 行); 互斥偏好集合相应惩罚值 $penaltySP_{mutex}$ 的初始化(第 2–3 行), 并按照惩罚值的降序排列; 设置初始最大惩罚值 $bound$ 为每个互斥偏好集合元素中最大的偏好惩罚值之和(第 4 行); 二维数组 $sumPenalty[m+1][bound+1]$用于记录过程约简的偏好惩罚值选择(第 5 行); 第 6 行初始化输出的约简偏好集合 SP_{opt}. 第 7–30 行为算法 2 的主循环, 每次迭代都会选出一组惩罚值之和小于等于 $bound$ 的偏好组合, 并基于 SMT 判断该组合是否可满足, 据此调整 n 和 SP_{opt}(第 25–28 行). 偏好组合的可满足性说明了偏好组合实现的可能性, 不可满足的偏好组合一定不会被规划解完全实现, 其中, 矛盾的偏好只能被规划解部分实现. 第 8–16 行为动态规划的过程, 在 $bound$ 的约束下寻找偏好组合的惩罚值极限. 遍历互斥偏好集合的每个元素, 在元素包含的所有偏好中做出选择: 如果所有偏好的惩罚值均大于剩余惩罚值极限, 则减少惩罚值极限(第 10–11 行), 继续下次迭代; 否则, 在前面已选择的偏好组合惩罚值的基础上, 挑选偏好的惩罚值加上累计惩罚值小于等于剩余惩罚值极限的最大惩罚值作为下次迭代的基础(第 12–13 行). 第 17–24 行根据最优惩罚值定位实际选择的偏好组合.

例 4(算法计算过程举例): 给定例 1 定义的规划问题. 根据步骤 1, 偏好 sp_2 作为单独的目标无法实现, 因此将其去除. 此时, 剩余的偏好集合为$\{sp_1,sp_3,sp_4,sp_5,sp_6\}$. 识别偏好公理后, 执行算法 1. 算法 1 经过内层循环计算, 第 1 次循环得到的互斥偏好集合$\{\{sp_1\},\{sp_3,sp_4\},\{sp_5\},\{sp_6\}\}$, 第 2 次循环得到的互斥偏好集合不再变化, 算法终止. 接着, 将算法 1 的结果输入给算法 2, 第 1 次循环得到的偏好组合为$\{sp_1,sp_3,sp_5,sp_6\}$(偏好惩罚和为 11), 第 2 次循环得到的偏好组合为$\{sp_3,sp_5,sp_6\}$(偏好惩罚和为 10), 第 3 次循环得到的偏好组合为$\{sp_4,sp_5,sp_6\}$(偏好惩罚和为 10). 由于 k=3, 算法 2 终止. 若 k=1, 得到的偏好组合就是算法 2 第 1 次循环的输出$\{sp_1,sp_3,sp_5,sp_6\}$. 若 k=2, 得到的偏好组合则是算法 2 第 2 次循环的输出$\{sp_3,sp_5,sp_6\}$. 两个偏好组合均不是最优组合, 只有当 k=3 时, 得到的偏好组合才是最优组合. 由此可见: 当 k 取值越大时, 获得最优组合的可能性越大. 然而, 随着 k 值的增加, 算法的计算时间也随之增长.

4 简单偏好的规划模型编码方法

2006 年, 第 5 届 IPC 举办了求解偏好的专项竞赛, 涌现出许多偏好规划器, 如 SGPlan[26]、OPTIC[27]、MIPS-XXL[28]等. 然而, 针对偏好的后续研究进展缓慢, 方法的创新和工具的更新较少. 经典规划方法近期取得了长足的发展, 高效的启发式算法和工具层出不穷, 如 Fast Downward[29]、LAMA[30]等. 因此, 本文尝试采用经典规划的方法求解偏好. 求解的过程主要涉及模型的转换, 即将带偏好的规划模型转换为经典规划模型. 本文称此过程为偏好模型编码.

4.1 从偏好模型到经典规划模型

给定一个带偏好的规划问题$\mathcal{P}=(P,A,I,HG,SP)$, 将$\mathcal{P}$转换为经典规划问题$\mathcal{P}'=(P',A',I',HG',D)$, 其中, D 为派生谓词集合, 其作用在后续文中给出解释. 目前, 大部分经典规划器均支持派生谓词, 是规划领域中不可或缺的公理形式. 然后, 直接利用经典规划器求解$\mathcal{P}'$. 所得的规划解需要进一步变形才能得到$\mathcal{P}$的规划解, 此步骤比较直观, 可以很容易地实现.

这里重点阐述从$\mathcal{P}$到$\mathcal{P}'$的转换过程, 本文称为$\mathcal{P}2\mathcal{P}'$. 经典规划器能够自动判断硬目标是否完成, 而对于偏好没有相应的机制. 因此, 转换后的模型需要在规划过程中显式判断偏好是否完成, 并将其作为新模型的硬目标之一. 规划的过程主要由 3 个模式组成(如图 3 所示), 即“规划模式”“进展模式”和“检查模式”. 在规划模式下执行一个原有规划动作, 然后进入进展模式. 在进展模式下, 随机执行两个动作之一, 其中一个动作执行后返回规划模式, 另一个动作执行后进入检查模式. 在检查模式下, 执行一组检查动作逐个判断偏好是否完成, 规划终止. 需要注意的是, 规划终止并不意味着找到规划解. 只有完成所有硬目标并且检查模式执行完毕, 才能得到规划解. 下面将详细说明转换的技术细节. 转换后的$\mathcal{P}'$定义如下.

- 派生谓词集合 D: $D=\{derived\_sp(\cdot)\triangleq sp|sp\in SP\}$. 对于任意 $sp\in SP$, 在规划模型中引入新的派生谓词 $derived\_sp(\cdot)\triangleq sp$, 其目的是简化模型的定义, 并在检查模式下判断偏好的满足情况, 以及便于偏好约

简中步骤(1)将偏好作为硬目标求解.

- 事实集合 P': $P'=P\cup\{derived\_sp(\cdot)|sp\in SP\}\cup\{planning(\cdot),progress(\cdot),done(\cdot)\}\cup\{check\_sp(\cdot)|sp\in SP\}$. 谓词 $planning(\cdot)$和 $progress(\cdot)$分别用于标记规划所处的模式为规划模式和进展模式. 谓词 $done(\cdot)$表示完成检查模式的标记. 由于检查偏好是否完成只能在最终状态进行, 一旦 $done(\cdot)$为真, 无论硬目标是否完成, 规划终止. 每次仅能执行一个偏好对应的检查动作.
- 动作集合 A': $A'=\{a'|a\in A\}\cup\{progress\_planning,progress\_check\}\cup\{sat\_sp,unsat\_sp|sp\in SP\}$. 每个规划动作 $a\in A$ 对应一个新动作 a', 其前置条件定义为 $pre(a')=pre(a)\cup\{planning(\cdot)\}$, 表示动作只能在规划模式下执行, 条件效果定义为 $eff(a')=eff(a)\cup\{\neg planning(\cdot),progress(\cdot)\}$, 表示动作执行后切换至进展模式, 动作权值与原动作相同 $cost(a')=cost(a)$. $\{progress_planning,progress_check\}$是两个新动作, 分别用于切换至规划模式和检查模式. 针对每个偏好 $sp\in SP$, 引入两个新动作 sat_sp 和 $unsat\_sp$. 当前状态满足 sp 时($derived_sp(\cdot)$成立), sat_sp 能够执行; 否则, $unsat\_sp$ 获得执行的资格. $progress\_planning$ 和 $progress\_check$ 以及 SP(假设 $SP=\{sp_1,\ldots,sp_n\}$)对应的动作组定义如下($1\leqslant i\leqslant n$).

$$progress\_planning=(\{progress(\cdot)\},\{\neg progress(\cdot),planning(\cdot)\},0)$$

$$progress\_check=(\{planning(\cdot)\},\{\neg planning(\cdot),check\_sp_1(\cdot)\},0)$$

$$sat\_sp_i=\left(\{check\_sp_i(\cdot),derived\_sp_i(\cdot)\},\left\{\neg check\_sp_i(\cdot),\begin{cases}check\_sp_{i+1}(\cdot), & \text{if } i<n\\ done(\cdot), & \text{if } i=n\end{cases}\right\},0\right)$$

$$unsat\_sp_i=\left(\{check\_sp_i(\cdot),\neg derived\_sp_i(\cdot)\},\left\{\neg check\_sp_i(\cdot),\begin{cases}check\_sp_{i+1}(\cdot), & \text{if } i<n\\ done(\cdot), & \text{if } i=n\end{cases}\right\},penalty(sp_i)\right)$$

$progress\_check$ 执行后, 会进入检查模式, 执行第 1 个偏好 sp_1 对应的检查动作. 接着, 其他剩余偏好对应的检查动作依次执行, 最后一个检查动作执行后, 将 $done(\cdot)$置为真. sat_sp_i 执行说明偏好满足, 惩罚值为 0; $unsat\_sp_i$ 执行说明偏好违背, 动作权值为偏好 sp_i 的惩罚值.

- 初始状态 I'和硬目标 HG': $I'=I\cup\{progress(\cdot)\}$, $HG'=HG\cup\{done(\cdot)\}$. 初始状态加入了谓词 $progress(\cdot)$, 表示初始模式为进展模式. 初始模式并没有置为规划模式, 原因在于初始状态有可能满足原始硬目标, 无须执行规划动作. 硬目标中加入了谓词 $done(\cdot)$, 表示必须完成偏好检查的规划解才符合要求.

图 3 偏好编码后的规划过程

例 5(偏好编码实例): 令$\mathcal{P}=(P,A,I,HG,SP)$为例 1 定义的规划问题. 根据上述转换方法, 偏好编码后的规划模型为$\mathcal{P}'=(P',A',I',HG',D)$, 具体定义见表 2. 引入 6 个派生谓词, 用于定义原有问题中的所有简单偏好. 事事集合中加入派生谓词$\{derived_sp_i(\cdot)|1\leqslant i\leqslant 6\}$、模式标记谓词 $planning(\cdot)$和 $progress(\cdot)$、偏好检查完成标记 $done(\cdot)$以及每个偏好的检查标记$\{check_sp_i(\cdot)|1\leqslant i\leqslant 6\}$. 每个原有规划动作的前置条件中均增加了 $planning(\cdot)$, 动作效果中均增加了$\neg planning(\cdot)$和 $progress(\cdot)$. 增加的动作除 $progress_planning(\cdot)$和 $progress_check(\cdot)$外, 每个偏好 sp_i 还对应两个动作 $sat_sp_i(\cdot)$和 $unsat\_sp_i(\cdot)$.

表 2 偏好编码实例

派生谓词集合 D	$\{derived\_sp_1(\cdot)\triangleq stored(goods_1,level_1),derived\_sp_2(\cdot)\triangleq stored(goods_1,level_2),$ $derived\_sp_3(\cdot)\triangleq stored(goods_2,level_1),derived\_sp_4(\cdot)\triangleq stored(goods_2,level_2),$ $derived\_sp_5(\cdot)\triangleq stored(goods_1,level_1)\rightarrow stored(goods_2,level_1),$ $derived\_sp_6(\cdot)\triangleq stored(goods_1,level_2)\rightarrow stored(goods_2,level_2)\}$

表 2 偏好编码实例(续)

事实集合 P'	$P\cup\{derived\_sp_i(\cdot)\|1\leqslant i\leqslant 6\}\cup\{planning(\cdot),progress(\cdot),done(\cdot)\}\cup\{check\_sp_i(\cdot)\|1\leqslant i\leqslant 6\}$	
动作集合 A'	$drive'(t,from,to)$ $pre(drive')=pre(drive)\cup\{planning(\cdot)\}$ $eff(drive')=eff(drive)\cup\{\neg planning(\cdot),progress(\cdot)\}$ $cost(drive')=1$	$load'(g,t,m,l_1,l_2,l_3,l_4)$ $pre(load')=pre(drive)\cup\{planning(\cdot)\}$ $eff(load')=eff(drive)\cup\{\neg planning(\cdot),progress(\cdot)\}$ $cost(load')=1$
	$unload'(g,t,d,l_1,l_2,l_3,l_4)$ $pre(unload')=pre(drive)\cup\{planning(\cdot)\}$ $eff(unload')=eff(drive)\cup\{\neg planning(\cdot),progress(\cdot)\}$ $cost(unload')=1$	$buy'(t,g,m,l_1,l_2,l_3,l_4)$ $pre(buy')=pre(drive)\cup\{planning(\cdot)\}$ $eff(buy')=eff(drive)\cup\{\neg planning(\cdot),progress(\cdot)\}$ $cost(buy')=1$
	$progress\_planning=(\{progress(\cdot)\},\{\neg progress(\cdot),planning(\cdot)\},0)$ $progress\_check=(\{planning(\cdot)\},\{\neg planning(\cdot),check\_sp_1(\cdot)\},0)$	
	$sat\_sp_i=\left(\{check\_sp_i(\cdot),derived\_sp_i(\cdot)\},\left\{\neg check\_sp_i(\cdot),\begin{cases}check\_sp_{i+1}(\cdot), & \text{if } i<6\\ done(\cdot), & \text{if } i=6\end{cases}\right\},0\right)$ $unsat\_sp_i=\left(\{check\_sp_i(\cdot),\neg derived\_sp_i(\cdot)\},\left\{\neg check\_sp_i(\cdot),\begin{cases}check\_sp_{i+1}(\cdot), & \text{if } i<6\\ done(\cdot), & \text{if } i=6\end{cases}\right\},penalty(sp_i)\right)$	
初始状态 I'	$I\cup\{progress(\cdot)\}$	
硬目标集合 HG'	$HG\cup\{done(\cdot)\}$	

4.2 理论证明

本节主要讨论偏好编码的复杂性、正确性和完备性等理论问题.

定理 1(转换复杂度). $\mathcal{P}2\mathcal{P}'$转换的时间复杂度与$\mathcal{P}$中动作集合和偏好集合的规模线性相关.

证明: 假设带偏好的规划问题$\mathcal{P}=(P,A,I,HG,SP)$, $\mathcal{P}$经过转换得到的规划模型为$\mathcal{P}'=(P',A',I',HG',D)$. $\mathcal{P}2\mathcal{P}'$转换主要包括派生谓词转换、事实转换、动作转换、初始状态转换和硬目标转换. 假设引入单个谓词的时间复杂度为常量, 各转换的时间复杂度分析如下.

- 派生谓词转换复杂度: 派生谓词定义的数量与偏好的数量一致, 因此其转换复杂度与偏好集合的规模线性相关, 即:

  $$O(|D|)=O(|\{derived\_sp(\cdot)\triangleq sp\,|\,sp\in SP\}|)=\sum_{sp\in SP}(|sp|+1)=\max(\{|sp|+1\,|\,sp\in SP\})O(|SP|).$$

  其中, $|sp|$为偏好公式 sp 包含的谓词数量.
- 事实转换复杂度: 新引入的谓词包括 3 种: 派生谓词、3 个标记谓词和检查偏好谓词. 其中, 派生谓词和检查偏好谓词的数量均与偏好集合的规模线性相关, 而 3 个标记谓词的转换复杂度为常量. 因此, 事实转换的复杂度为

  $$O(|\{derived\_sp(\cdot)|sp\in SP\}\cup\{planning(\cdot),progress(\cdot),done(\cdot)\}\cup\{check\_sp(\cdot)|sp\in SP\}|)=2O(|SP|).$$
- 动作转换复杂度: 在规划模式中, 在每个原有动作中加入了 3 个谓词, 因此, 规划模式下动作的转换复杂度为 $3O(|A|)$. 在偏好检查动作组中, 每个偏好对应两个动作, 每个动作有 4 个谓词, 因此, 偏好模式下的动作转换复杂度为 $8O(|SP|)$. 负责模式转换的两个动作的转换复杂度为常量.
- 初始状态和硬目标转换复杂度: 初始状态和硬目标均引入了一个额外的谓词, 故转换复杂度为常量.

综上, $\mathcal{P}2\mathcal{P}'$转换的复杂度为 $\max(\{|sp|+1|sp\in SP\})O(|SP|)+2O(|SP|)+8O(|SP|)+3O(|A|)$. 故定理成立.

定理 2(正确性). 若$\mathcal{P}'$是带偏好的规划问题$\mathcal{P}$经过$\mathcal{P}2\mathcal{P}'$转换得到的规划问题, 则$\mathcal{P}'$的每个规划解对应$\mathcal{P}'$的一个规划解.

证明: 令$\mathcal{P}=(P,A,I,HG,SP)$且 $SP=\{sp_1,\ldots,sp_n\}$, 经过$\mathcal{P}2\mathcal{P}'$转换得到的模型为$\mathcal{P}'=(P',A',I',HG',D)$. 首先证明$\mathcal{P}'$的规划解一定形如$\pi_1=\langle progress_check,check_sp_1,\ldots,check_sp_n\rangle$或$\pi_2=\langle progress_planning,a_1,\ldots,progress_planning,a_m,progress_check,check_sp_1,\ldots,check_sp_n\rangle$, $a_i\in A'$, $1\leqslant i\leqslant m$. 已知硬目标中包含 $done(\cdot)$, 说明规划解的后缀一定形如π_1. 若初始状态即满足除 $done(\cdot)$以外的其他硬目标 $HG\backslash\{done(\cdot)\}$, 则$\pi_1$即为规划解; 否则, 需在

π_1 前执行一些规划动作. 在此情况下, 已知初始状态 *progress*(·)成立, 则第 1 个动作一定执行 *progress_planning*, 然后执行一个规划动作, 以此类推, *progress_planning* 与规划动作交替执行, 完成硬目标后执行π_1. 最终的规划解形如π_2.

不失一般性, 假设$\mathcal{P}'$的规划解为π_2. 容易得知, 引入的额外动作不会改变原有问题中的谓词真值. 在π_2中去除这些动作, 剩余的动作序列为$\pi_3=\langle a_1,\ldots,a_m\rangle$. 再将这些动作还原至原来的规划动作, 去除前置条件和条件效果中加入的新谓词, 得到的动作序列为 $\pi'_3=\langle a'_1,\ldots,a'_m\rangle$, $a'_i\in A$, $1\leqslant i\leqslant m$. 容易得知, π'_3能够顺序执行, 并完成除 *done*(·)外的其他硬目标 $HG\backslash\{done(\cdot)\}$. 因此, π'_3是$\mathcal{P}$的一个规划解. 定理得证.

定理 3(完备性). 若$\mathcal{P}'$是带偏好的规划问题$\mathcal{P}$经过$\mathcal{P}2\mathcal{P}'$转换得到的规划问题, 则$\mathcal{P}$的每个规划解对应$\mathcal{P}'$的一个规划解.

证明: 令$\mathcal{P}=(P,A,I,HG,SP)$, 且 $SP=\{sp_1,\ldots,sp_n\}$, 经过$\mathcal{P}2\mathcal{P}'$转换得到的模型为$\mathcal{P}'=(P',A',I',HG',D)$. 假设$\pi=\langle a_1,\ldots,a_m\rangle$是$\mathcal{P}$的规划解. 先将$\pi$中所有动作替换为$\mathcal{P}'$中的对应动作, 可得$\pi'=\langle a'_1,\ldots,a'_m\rangle$, 其中, $a'_i\in A'$为 a_i 转换后的动作, $1\leqslant i\leqslant m$. 然后, 在每个动作前加入动作 *progress_planning*, 得到$\pi''=\langle progress_planning, a'_1,\ldots, progress_planning, a'_m\rangle$. 易知, π''可以完成硬目标 *HG*. 为了完成$\mathcal{P}'$中的硬目标 *done*(·), 将π''加入后缀$\pi'''=\langle progress_check, check_sp_1,\ldots, check_sp_n\rangle$. 因此, $\langle\pi'',\pi'''\rangle$能够完成硬目标 HG', 它是$\mathcal{P}'$的一个规划解. 定理得证.

5 实验分析

本节主要通过实验说明偏好约简和偏好编码方法的有效性, 选取的基准测试集来自第 5 届 IPC. 实验采用的偏好规划器为 SGPlan, 经典规划器为 FastDownward (FD). SGPlan 是表现最好的偏好规划器, 在第 5 届 IPC 的可满足规划竞赛(satisficing planning)中荣获第一名. FD 是经典规划领域最为知名的规划器, 许多参加 IPC 竞赛的规划器都吸取了 FD 的优点, 有的甚至直接继承了它的代码. 2021 年 12 月, Fast Downward 发布了最新的 Release 版本, 供学术界和工业界使用(https://www.fast-downward.org/Releases). 我们实现了偏好约简算法以及偏好编码方法. 实验运行的平台配置为 Ubuntu 16.04 操作系统, Intel i7 CPU, 8 GB 内存.

实验求解的具体规划问题为 TPP 和 trucks, 两个问题中存在偏好互斥和关联的情况. 实验结果的对比标准有两个: 规划解的质量和求解时间. 规划解质量又分为两部分——动作开销和偏好惩罚. 求解时间说明方法的求解效率. 动作开销和偏好惩罚越小, 说明规划解质量越高. 实验过程中, 算法 2 的配置参数 k 取 3. TPP 问题无须实现任何硬目标, 仅对偏好进行定义, 而 trucks 问题既包含硬目标, 也包含偏好. 每个问题都包含 20 个实例, 且求解时间限制为 30 min.

第 1 个实验运行偏好规划器 SGPlan, 验证偏好约简的有效性, 表 3 列出了相应的实验结果. 实例的名称通过行名 p**的形式标记. “*time*”表示规划时间, “*cost*”表示规划解的动作开销, “*penalty*”表示规划解的偏好惩罚. “SGPlan (before)”和“SGPlan (after)”分别代表偏好集合约简前后的运行结果. “*cost*%” (“*penalty*%”)表示 SGPlan (after)相对于 SGPlan (before)运行每个示例的动作开销(偏好惩罚)提升比例. 总的来看, 求解约简后的偏好集合令 SGPlan 各方面的指标均得到显著提升. 尤其对于所有实例, 都获得了更低的偏好惩罚, 且 TPP 问题的规划解均为最优解. 对于 TPP 问题的绝大多数实例(17 个)和 trucks 问题的大部分实例(13 个), 均获得了较低的动作开销, 剩余实例动作开销增加幅度很小, 其增长的目的也是为了降低偏好惩罚. 平均来讲, TPP 问题的动作开销降低了 0.13%, 偏好惩罚降低了 10.18%; trucks 问题的动作开销增加了 1.02%, 偏好惩罚降低了 11.57%. 换句话说, 偏好约简方法的优势主要在于降低偏好惩罚. 在规划时间方面, SGPlan (after)求解 TPP 问题的时间相比 SGPlan (after)有明显的降低, 而求解 trucks 问题并未如预期的一样约简偏好后降低规划时间,这点还需要进一步对 SGPlan 的启发式算法进行深入分析.

第 2 个实验运行经典规划器 FD, 验证偏好编码方法和偏好约简方法相结合的有效性, 求解的规划问题仍然是 TPP 和 trucks. 表 4 列出了相应的实验结果. “FD (before)”和“FD (after)”分别代表仅采用偏好编码、结合偏好编码和偏好约简的运行结果. 类似于 SGPlan 的运行结果, 求解 TPP 问题获得的所有规划解均为最优解. 与 FD (before)相比, FD (after)在偏好惩罚方面具备较大优势, 后者能够将 TPP 问题的偏好惩罚平均降低 1/4 以

上(28.59%), 将 trucks 问题的偏好惩罚平均降低 2.16%. 这从另外一个角度说明, 没有偏好约简而直接使用偏好编码的效果较差. 由于 TPP 问题的特殊性(无硬目标), 导致偏好约简前后的动作开销差距非常明显. FD (before)的动作开销比较低, 因此实现的偏好较少. FD (before)求解 TPP 问题有 3 个实例甚至没有执行任何规划动作. 这样虽然获得的动作开销最低, 但是相应的惩罚也较高. 偏好约简前, 待求解的偏好集合规模较大, FD (before)难以权衡偏好的实现情况, 因此求得的规划解动作较少, 即未实现更多的偏好, 仅令动作开销保持较低. 偏好约简后, 偏好集合规模大大减少, 使得 FD (after)有能力在求解的过程中尽量满足偏好. FD (before)在求解 trucks 时此问题不突出, 这是因为有硬目标的存在, 且硬目标与偏好存在一定的关联. 在求解时间方面, FD (after)比 FD (before)表现更好. 究其原因仍然是, 偏好约简使得偏好集合规模变小. 与 SGPlan (after)相比, FD (after)获得的动作开销更优. 也就是说, 在惩罚权值相同的情况下, FD 能够优化动作开销, 这是由 FD 本身的特点决定的. 作为经典规划器, FD 通常求解的过程总是寻求更优的动作开销. 此外, FD 在规划的最后才检查偏好的完成情况, 相当于在求解过程中没有给启发式算法足够的信息求解偏好, 这也是 FD 求解偏好惩罚的结果略差于 SGPlan 的原因.

表 3 偏好规划器 SGPlan 的实验结果

规划问题		SGPlan (before)			SGPlan (after)			before vs. after	
		time	cost	penalty	time	cost	penalty	cost%	penalty%
TPP	p01	**0**	**25**	**16**	**0**	**25**	**16**	0	0
	p02	**0**	**20**	**24**	**0**	**20**	**24**	0	0
	p03	**0**	**30**	**29**	**0**	**30**	**29**	0	0
	p04	**0**	**30**	**35**	**0**	**30**	**35**	0	0
	p05	**0.02**	**90**	79	**0.02**	**90**	**39**	0	50.63
	p06	**0.04**	**70**	**101**	**0.04**	**70**	**101**	0	0
	p07	**0.02**	**115**	**100**	**0.02**	**115**	**100**	0	0
	p08	**0.02**	**135**	**105**	**0.02**	**135**	**105**	0	0
	p09	**0.06**	**230**	**205**	0.09	**230**	**205**	0	0
	p10	0.1	220	282	**0.08**	**190**	**261**	13.64	7.45
	p11	8.15	**205**	295	**6.14**	265	**173**	–29.27	41.36
	p12	0.27	**220**	308	**0.2**	**220**	**215**	0	30.19
	p13	0.5	**255**	**750**	**0.48**	**255**	**750**	0	0
	p14	**29.29**	**275**	821	34.76	280	**764**	–1.82	6.94
	p15	0.69	**325**	837	**0.66**	**325**	**719**	0	14.1
	p16	87.61	280	941	**67.36**	**275**	**764**	1.79	18.81
	p17	22.92	**490**	**1 633**	**14.24**	**490**	**1 633**	0	0
	p18	6.41	**540**	1 576	**4.95**	590	**1 497**	–9.26	5.01
	p19	14.88	575	1 829	**9.74**	**515**	**1 603**	10.43	12.36
	p20	128.87	615	1 890	**2.08**	**510**	**1 573**	17.07	16.77
	Avg.	–	–	–	–	–	–	**0.13**	**10.18**
trucks	p01	**0**	**14**	37	**0**	**14**	**30**	0	18.92
	p02	**0**	**18**	100	**0**	**18**	**84**	0	16
	p03	**0**	**24**	53	**0**	**24**	**45**	0	15.09
	p04	**0**	**28**	150	**0**	**28**	**132**	0	12
	p05	**0**	**33**	151	**0**	**33**	**132**	0	12.58
	p06	**0.02**	**39**	266	**0.02**	**39**	**240**	0	9.77
	p07	0.66	**38**	998	**0.27**	**38**	**918**	0	8.02
	p08	**0.1**	**39**	1 655	0.76	**39**	**1 520**	0	8.16
	p09	**0.16**	**42**	2 239	1.48	**42**	**2 079**	0	7.15
	p10	**0.2**	**58**	2 049	1.55	**58**	**1 932**	0	5.71
	p11	**0.13**	**48**	536	0.16	50	**478**	–4.17	10.82
	p12	**0.2**	55	677	0.3	**53**	**620**	3.64	8.42
	p13	**0.19**	**58**	686	0.29	60	**618**	–3.45	9.91
	p14	**0.31**	**60**	935	0.41	62	**792**	–3.33	15.29
	p15	**0.78**	**70**	1 052	1.1	71	**978**	–1.43	7.03
	p16	**1.42**	**73**	1 454	1.92	75	**1 280**	–2.74	11.97
	p17	4.17	75	1 820	**3.84**	**74**	**1 630**	1.33	10.44
	p18	**3.58**	77	2 060	5.29	**76**	**1 550**	1.3	24.76
	p19	**4.8**	**82**	1 692	12.95	86	**1 490**	–4.88	11.94
	p20	**6.98**	**90**	2 230	13.83	96	**2 065**	–6.67	7.4
	Avg.	–	–	–	–	–	–	**–1.02**	**11.57**

表 4 经典规划器 FastDownward 的实验结果

规划问题		FD (before)			FD (after)			before vs. after	
		time	cost	penalty	time	cost	penalty	cost%	penalty%
TPP	p01	0.16	**18**	**16**	**0.04**	**18**	**16**	0	0
	p02	0.14	**14**	**24**	**0**	**14**	**24**	0	0
	p03	2.14	**20**	**29**	**0.78**	**20**	**29**	0	0
	p04	60.88	**20**	**35**	**0.72**	**20**	**35**	0	0
	p05	55.08	**11**	102	**0.3**	68	**39**	518.18	61.76
	p06	99.6	**11**	116	**75.66**	48	**101**	336.36	12.93
	p07	33.12	**8**	133	**3.22**	80	**100**	900	24.81
	p08	18.18	**8**	148	**14.28**	94	**105**	1 075	29.05
	p09	75.26	**8**	339	**53.64**	202	**205**	2 425	39.53
	p10	124.58	**8**	370	**1.26**	146	**261**	1 725	29.46
	p11	26.96	**5**	402	**9.18**	214	**173**	4 180	56.97
	p12	39.44	**5**	433	**5.5**	176	**215**	3 420	50.35
	p13	27.5	**5**	944	**22.26**	220	**750**	4 300	20.55
	p14	32.08	**5**	1 007	**18.92**	223	**764**	4 360	24.13
	p15	42.5	**5**	1 070	**13.68**	289	**719**	5 680	32.8
	p16	52.06	**5**	1 133	**26.72**	230	**764**	4 500	32.57
	p17	**2.8**	**0**	2 413	21.96	434	**1 633**	*	32.32
	p18	**4.9**	**0**	2 540	48.14	514	**1 497**	*	41.06
	p19	**8.94**	**10**	2 665	9.88	476	**1 603**	4 660	39.85
	p20	**4.3**	**0**	2 794	23.28	474	**1 573**	*	43.7
	Avg.	–	–	–	–	–	–	**2 239.97**	**28.59**
trucks	p01	**0**	**13**	**30**	**0**	**13**	**30**	0	0
	p02	0.12	**17**	**84**	**0**	**17**	**84**	0	0
	p03	0.02	**20**	**45**	**0.02**	**20**	**45**	0	0
	p04	3.5	**23**	**132**	**1.3**	**23**	**132**	0	0
	p05	**1.28**	**25**	182	4.38	**25**	**132**	0	27.47
	p06	10.56	34	**240**	**3.64**	**33**	**240**	2.94	0
	p07	**0.7**	38	**918**	0.9	**36**	**918**	5.26	0
	p08	**0.88**	44	**1 520**	76.46	**38**	**1 520**	13.64	0
	p09	1.22	49	**2 079**	**0.18**	**42**	**2 079**	14.29	0
	p10	**3.38**	52	1 956	59.3	**51**	**1 932**	1.92	1.23
	p11	**0.8**	50	486	5.5	**42**	**478**	16	1.65
	p12	**0.9**	55	640	23.88	**47**	**620**	14.55	3.13
	p13	1.9	63	626	**1.02**	**52**	**618**	17.46	1.28
	p14	**1.5**	59	800	79.82	**54**	**792**	8.47	1
	p15	**2**	62	994	16.14	**59**	**978**	4.84	1.61
	p16	16.46	82	1 360	**4.1**	**80**	**1 315**	2.44	3.31
	p17	7.1	**74**	1 680	**1.78**	78	**1 690**	–5.13	0.6
	p18	5.48	**82**	1 620	**1.56**	85	**1 590**	–3.66	1.85
	p19	5.92	**83**	**1 535**	**3.02**	87	**1 535**	–4.82	0
	p20	19.54	97	**2 165**	**3.56**	**89**	**2 165**	8.25	0
	Avg.	–	–	–	–	–	–	**4.82**	**2.16**

6 结论与展望

本文针对智能规划中的简单偏好进行了研究, 采用 SMT 技术对待求解的偏好集合进行约简, 并提出了一种偏好的编码方法, 将带偏好的规划模型转换为经典规划模型. 实验结果表明, 约简后的偏好集合更有利于规划器进行求解, 获得的规划解质量更优. 本文提出的偏好编码方法使得经典规划器也可直接求解带偏好的规划问题, 其灵活性和可扩展性较强.

未来将开展复杂偏好的约简方法研究工作. 复杂偏好与简单偏好不同, 前者描述的是规划过程需满足的约束, 而后者仅对最终状态进行限制. PDDL 3.0 版本定义的复杂偏好理论上是时序逻辑 LTL_f[31]的子集. 例如, (always ϕ)要求所有可达状态均满足ϕ. 目前存在一些 LTL_f 公式的可满足性求解工具(Aalta-finite[32]、ltl2sat[33]), 以此为基础, 可尝试研究类似本文的复杂偏好的约简方法. 利用求解工具得到复杂偏好的互斥集合划分以及和组合. 然而, 偏好公理仍然只能针对单一状态进行约束. 为了进一步精确划分复杂偏好, 需提取规划模型中具备状态序列描述能力的偏好公理.

References:

[1] Malik G, Dana SN, Paolo T. Automated Planning—Theory and Practice. Elsevier, 2004.

[2] Gerevini A, Long D. Preferences and soft constraints in PDDL3. In: Proc. of the ICAPS Workshop on Planning with Preferences and Soft Constraints. 2006. 46−53.

[3] Baier JA, McIlraith SA. Planning with preferences. AI Magazine, 2008, 29(4): 25−36. [doi: 10.1609/aimag.v29i4.2204]

[4] Son TC, Pontelli E. Planning with preferences using logic programming. Theory and Practice of Logic Programming, 2006, 6(5): 559−607. [doi: 10.1017/S1471068406002717]

[5] Baier JA, Bacchus F, McIlraith SA. A heuristic search approach to planning with temporally extended preferences. In: Veloso MM, ed. Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2007). 2007. 1808−1815.

[6] Sohrabi S, Baier JA, McIlraith SA. HTN planning with preferences. In: Boutilier C, ed. Proc. of the 21st Int'l Joint Conf. on Artificial Intelligence (IJCAI 2009). 2009. 1790−1797.

[7] Wright B, Mattmüller R, Nebel B. Compiling away soft trajectory constraints in planning. In: Thielscher M, Toni F, Wolter F, eds. Proc. of the 16th Int'l Conf. (KR 2018). 2018. 474−483.

[8] Percassi F, Gerevini AE. On compiling away PDDL3 soft trajectory constraints without using automata. In: Benton J, Lipovetzky N, Onaindia E, *et al*., eds. Proc. of the 29th Int'l Conf. on Automated Planning and Scheduling (ICAPS 2019). 2019. 320−328.

[9] Bonassi L, Gerevini AE, Percassi F, *et al*. On planning with qualitative state-trajectory constraints in PDDL3 by compiling them away. In: Biundo S, Do M, Goldman R, *et al*., eds. Proc. of the 31st Int'l Conf. on Automated Planning and Scheduling (ICAPS 2021). 2021. 46−50.

[10] Mattmüller R, Geißer F, Wright B, *et al*. On the relationship between state-dependent action costs and conditional effects in planning. In: McIlraith SA, Weinberger KQ, eds. Proc. of the 32nd AAAI Conf. on Artificial Intelligence (AAAI 2018). 2018. 6237−6245.

[11] Bofill M, Espasa J, Villaret M. Relaxed exists-step plans in planning as SMT. In: Sierra C, ed. Proc. of the 26th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2017). 2017. 563−570. [doi: 10.24963/ijcai.2017/79]

[12] Bit-Monnot A, Leofante F, Pulina L, *et al*. SMT-based planning for robots in smart factories. In: Wotawa F, Friedrich G, Pill I, *et al*., eds. Proc. of the 32nd Int'l Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2019). 2019. 674−686. [doi: 10.1007/978-3-030-22999-3_58]

[13] Sebastiani R, Tomasi R. Optimization in SMT with LA(Q) cost functions. In: Gramlich B, Miller D, Sattler U, eds. Proc. of the 6th Int'l Joint Conf. on Automated Reasoning (IJCAR 2012). 2012. 484−498. [doi: 10.1007/978-3-642-31365-3_38]

[14] Chen L, Wang YJ, Wu JZ, *et al*. Rate-monotonic optimal design based on tree-like linear programming search. Ruan Jian Xue Bao/Journal of Software, 2015, 26(12): 3223−3241 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4853.htm [doi: 10.13328/j.cnki.jos.004853]

[15] Blackham B, Liffiton MH, Heiser G. Trickle: Automated infeasible path detection using all minimal unsatisfiable subsets. In: Proc. of the 20th IEEE Real-time and Embedded Technology and Applications Symp. (RTAS 2014). 2014. 169−178. [doi: 10.1109/RTAS.2014.6926000]

[16] Eldib H, Wang C, Taha M, *et al*. Quantitative masking strength: Quantifying the power side-channel resistance of software code. IEEE Trans. on Computer-aided Design of Integrated Circuits and Systems, 2015, 34(10): 1558−1568. [doi: 10.1109/TCAD.2015.2424951]

[17] Leroy X. Formal verification of a realistic compiler. Communications of the ACM, 2009, 52(7): 107−115. [doi: 10.1145/1538788.1538814]

[18] Klein G. Operating system verification—An overview. Sadhana, 2009, 34(1): 27−69. [doi: 10.1007/s12046-009-0002-4]

[19] de Moura LM, Bjørner NS. Z3: An efficient SMT solver. In: Ramakrishnan CR, Rehof J, eds. Proc. of the 14th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008). 2008. 337−340. [doi: 10.1007/978-3-540-78800-3_24]

[20] Barrett CW, Conway CL, Deters M, Operating system verification—An overview. CVC4. In: Gopalakrishnan G, Qadeer S, eds. Proc. of the 23rd Int'l Conf. on Computer Aided Verification (CAV 2011). 2011. 171−177. [doi: 10.1007/978-3-642-22110-1_14]

[21] Dutertre B. Yices 2.2. In: Biere A, Bloem R, eds. Proc. of the 26th Int'l Conf. on Computer Aided Verification (CAV 2014). 2014. 737−744. [doi: 10.1007/978-3-319-08867-9_49]

[22] Cimatti A, Griggio A, Schaafsma BJ, *et al*. The MathSAT5 SMT solver. In: Piterman N, Smolka SA, eds. Proc. of the 19th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2013). 2013. 93−107. [doi: 10.1007/978-3-642-36742-7_7]

[23] Brummayer R, Biere A. Boolector: An efficient SMT solver for bit-vectors and arrays. In: Kowalewski S, Philippou A, eds. Proc. of the 15th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009). 2009. 174−177. [doi: 10.1007/978-3-642-00768-2_16]

[24] Godefroid P, de Halleux J, Nori AV, *et al*. Automating software testing using program analysis. IEEE Software, 2008, 25(5): 30−37. [doi: 10.1109/MS.2008.109]

[25] Beyer D, Keremoglu ME. CPAchecker: A tool for configurable software verification. In: Gopalakrishnan G, Qadeer S, eds. Proc. of the 23rd Int'l Conf. on Computer Aided Verification (CAV 2011). 2011. 184−190. [doi: 10.1007/978-3-642-22110-1_16]

[26] Hsu CW, Wah WB, Huang R, *et al*. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In: Veloso MM, ed. Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2007). 2007. 1924−1929.

[27] Benton J, Coles AJ, Coles A. Temporal planning with preferences and time-dependent continuous costs. In: McCluskey L, Williams BC, Silva JR, *et al*., eds. Proc. of the 22nd Int'l Conf. on Automated Planning and Scheduling (ICAPS 2012). 2012. 2−10.

[28] Edelkamp S, Jabbar S, Nazih M. Large-scale optimal PDDL3 planning with MIPS-XXL. In: Proc. of the 5th Int'l Planning Competition Booklet (IPC 2006). 2006. 28−30.

[29] Helmert M. The fast downward planning system. Journal of Artificial Intelligence Research, 2006, 26: 191−246. [doi: 10.1613/jair.1705]

[30] Richter S, Westphal M. The LAMA planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research, 2010, 39: 127−177. [doi: 10.1613/jair.2972]

[31] Li J, Zhang L, Pu G, *et al*. LTLf satisfiability checking. In: Schaub T, Friedrich G, O'Sullivan B, eds. Proc. of the 21st European Conf. on Artificial Intelligence (ECAI 2014). IOS, 2014. 513−518. [doi: 10.3233/978-1-61499-419-0-513]

[32] Li J, Pu G, Zhang Y, *et al*. SAT-based explicit LTLf satisfiability checking. Artificial Intelligence, 2020, 289: Article No.103369. [doi: 10.1016/j.artint.2020.103369]

[33] Fionda V, Greco G. The complexity of LTL on finite traces: Hard and easy fragments. In: Schuurmans D, Wellman MP, eds. Proc. of the 30th AAAI Conf. on Artificial Intelligence (AAAI 2016). 2016. 971−977.

附中文参考文献:

[14] 陈力, 王永吉, 吴敬征, 等. 基于树状线性规划搜索的单调速率优化设计. 软件学报, 2015, 26(12): 3223−3241. http://www.jos.org.cn/1000-9825/4853.htm [doi: 10.13328/j.cnki.jos.004853]

陆旭(1985－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为智能规划, 模型检测, 程序验证.

于斌(1990－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为模型检测, 运行时验证.

段振华(1948－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为时序逻辑, 形式化方法, 高可信嵌入式系统.

王德奎(1991－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为 FPGA EDA 算法优化, 脑电信号处理.

陈蠡(1978－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为软件安全检测, 学习安全与隐私保护.

崔进(1989－), 女, 博士, 讲师, CCF 专业会员, 主要研究领域为可信软件, 实时系统, 大数据计算.