基于动态约束满足框架的强表达时态规划算法

刘越畅
(嘉应学院计算机学院 梅州 514015)

摘 要 智能规划已经成为人工智能领域最热门的研究主题之一。近年来,智能规划在现实领域的应用越来越广泛,这对规划器的处理能力和效率提出了很大的挑战。以一类强表达时态规划——基于约束区间规划为研究对象,基于动态约束满足框架设计和实现了一个基于约束区间的规划算法LP-TPOP;对算法的可靠性和完备性进行了证明;最后以一个规划实例演示了算法的运行过程。

关键词 智能规划,强表达时态规划,动态约束满足问题,算法

中图法分类号 TP181 **文献标识码** A

Expressive Temporal Planning Algorithm under Dynamic Constraint Satisfaction Framework

LIU Yue-chang
(School of Computer Science, Jiaying University, Meizhou 514015, China)

Abstract AI planning has been the key research topic in artificial intelligence(AI) community. In recent years, AI planning technology has been applied to solve many real-world problems, which gives a challenge for AI planner both in handling power and efficiency. This paper studied a kind of expressive temporal planning paradigm——constraint based interval(CBI) planning. Based on the dynamic constraint satisfaction problem framework, the author designed a new CBI algorithm named LP-TPOP. The paper gave the proof of soundness and completeness for LP-TPOP, and algorithm demonstration for a CBI planning example.

Keywords AI planning, Expressive temporal planning, Dynamic constraint satisfaction problem, Algorithm

1 引言

人工智能规划(或称自动规划,automated planning)是人们在对因果推理、动作推理以及时态推理研究的基础上发展起来的。早期的智能规划研究是在问题解决(problem solving)的框架下,借助定理证明的纯逻辑方法实现的[1],这导致其表达能力和求解效率相当低下,使得智能规划只停留在理论研究上,难以解决实际中的问题。自STRIPS规划方法[5]提出以来,智能规划的研究得到了长足的进展,产生了许多不同类型的规划算法,如基于搜索的规划方法、部分序规划、图规划、层次任务网规划方法、SAT规划、基于模型检测的规划等。智能规划的发展也体现在规划所能处理的世界要素和特性越来越丰富,如时态、资源、外部事件、自动过程、不确定因素等[2]。

实际的规划问题中,动作间往往包含着复杂的交互关系,并且动作的执行往往不是瞬时的,而是持续一段时间的。为了表示这些实际的规划领域,需要将时间因素显式地表示在动作描述中。将时间因素加入到动作描述中有多种形式。David E. Smith在他的时态规划器TGP(Temporal Graph Planning)中使用了一个时态动作模型——TGP模型[3]:每个动作都有一个持续时间;该动作执行前所有的前提条件都必须成立,且在动作执行过程中前提条件也必须保持成立;所有效果在动作执行结束瞬间开始成立。这样的动作模型也称为黑盒模型(black-box model)。2002年的国际规划比赛是第一届强调时态规划的比赛,这届比赛中使用PDDL2.1[4]作为标准时态规划语言。PDDL2.1是对PDDL1.0[6]的扩充。相对于TGP模型,PDDL2.1引入了丰富的时态元素表示。除了允许每个动作有一个持续时间以外,PDDL2.1还可以表达3类不同的前提条件(动作执行开始时刻必须成立的条件(at-start条件)、动作结束的时刻必须成立的条件(at-end条件)和动作执行过程中必须始终成立的条件(overall条件))以及两类不同的效果(动作开始执行时刻开始成立的效果(at-start效果)、动作执行结束时刻开始成立的效果(at-end效果))。这些丰富的时态特征使得PDDL2.1可以表达动作间存在的复杂的交互关系的规划领域;然而,PDDL2.1中的over-all条件和at-end条件破坏了规划领域的马尔可夫特性[7]。Cushing, Mausam和Kambhampati等人将PDDL2.1时态规划问题分为简单时态规划(simple temporal planning)和强表达时态规划问题(expressive temporal planning)[8]。在对PDDL2.1评论的文章中,Smith提出了一种更为丰富的动作模型[9],这种动作模型允许表示在动作持续时间内的任意区间的前提条件,以及任意区间内确保有效的效果命题。Smith称这种动作模

到稿日期:2011-07-26 返修日期:2011-11-24 本文受梅州市科学技术局,梅州市科技计划项目(梅市科(2011)3号),嘉应学院联合自然科学研究项目(2010KJA06),嘉应学院科研启动经费项目资助。

刘越畅(1979—),男,博士,讲师,主要研究方向为知识工程、智能规划和调度、时态推理,E-mail:ycliu@jyu.edu.cn。

型为基于约束的区间(Constraint Based Interval, CBI)模型[10]。以这种动作模型为基础,他提出了一种通过区间时态约束管理来整合规划和调度的算法框架。CBI 模型比 PDDL2.1 具有更丰富的时态表达元素,Sara Bernardini 证明了 PDDL2.1 规划问题可以等价转化为 CBI 规划问题[15]。很自然,CBI 规划也是强表达时态规划语言。在 CBI 规划基础上,Smith 等人进一步提出了 NDDL 规划语言,并开发了 Europa 及 Europa2 规划器[11,17,18]。

本文提出一个基于动态约束满足求解框架[12]的 CBI 规划算法 LP-TPOP。与 Smith 的 CBI 算法和 Europa/Europa2 类似,LP-TPOP 采用的是动态约束满足求解技术;然而,与 CBI 规划方法中使用简单时态问题(Simple Temporal Problem, STP)不同的是,LP-TPOP 采用表达能力更强的析取时态问题(Disjunctive Temporal Problem, DTP)[13,19]作为时态表示和推理框架(使用 DTP 求解器 Graph-DTP[19]),从而具有更合理的计算量分布[20]。与 Europa/Europa2 规划算法使用以状态变量为中心的推理框架不同的是,LP-TPOP 算法采取以动作为中心的表达和推理框架,从而具有较小的分支因子。

本文第 2 节给出动态约束满足问题和 CBI 规划的相关概念定义;第 3 节详细描述了 LP-TPOP 算法细节,并给出太空飞行器的实例演示以及算法可靠性、完备性;最后总结本文工作。

2 基本定义

2.1 约束满足问题(CSP)和动态约束满足问题(DCSP)

定义 1 约束满足问题是个三元组 $P=\langle X,D,C\rangle$,其中,

$X=\{x_1,x_2,\cdots,x_n\}$是一个变量集合;

$D=\{D_1,D_2,\cdots,D_n\}$是对应于 X 的(有限)取值域组成的集合,即 x_i 的取值域为(有限)集合 D_i;

$C=\{c_1,c_2,\cdots,c_m\}$,其中,c_i 称为对应于变量 $x_{i,1},x_{i,2},\cdots,x_{i,k}(k\geqslant 1)$的约束,有 $c_i=\{\langle x_j,d_j\rangle \mid x_j\in X,d_j\in D_j\}$表示非法的变量赋值元组。给定一个约束可满足问题 P 的解是一个解释 $s=\{x_i=d_i \mid i=1,2,\cdots,n,d_i\in D_i\}$,且 s 满足 P 中所有约束。

定义 2 一个动态约束满足问题是一个 CSP:$P=\langle X,D,C\rangle$,其中 $C=C_0\cup C_1\cup C_2\cup C_3$:

$C_0=\{active(x_{a1}),active(x_{a2}),\cdots,active(x_{aq}) \mid x_{ai}\in X\}$;

$C_1=\{c_1,c_2,\cdots,c_m\}$,$c_i=\{\langle x_i,d_i\rangle \mid x_i\in X,d_i\in D_i\}$表示非法的变量赋值元组;

$C_2=\{ac_1,ac_2,\cdots,ac_m\}$,$ac_i=\{x_{i1}=d_{i1}\wedge x_{i2}=d_{i2}\wedge,\cdots,\wedge x_{ik}=d_{ik}\rightarrow active(x_j) \mid i=1,2,\cdots,l\}\cup\{x_{i1}=d_{i1}\wedge x_{i2}=d_{i2}\wedge,\cdots,\wedge x_{ik}=d_{ik}\rightarrow inactive(x_j) \mid i=1,2,\cdots,h\}$;

$C_3=\{active(x)\Leftrightarrow\neg inactive(x) \mid x\in X\}$。

其中,C_0 称为初始激活变量约束集,C_1 为原始约束集,C_2 为激活约束集,C_3 为激活性约束。

2.2 CBI 规划相关定义

定义 3 时态限定表达式(Temporally Qualified Expression, TQE)是形如$\langle t_s,p(l_1,l_2,\cdots,l_k),t_e\rangle$的三元组,其中,$p(l_1,l_2,\cdots,l_k)$是一个以 $l_1,l_2,\cdots,l_k$ 为参数的谓词符号,$t_s,t_e\in\mathbf{Z}(t_s\leqslant t_e)$分别表示 $p(l_1,l_2,\cdots,l_k)$成立的起始时间点变量符号和结束时间点变量符号。时态限定命题(Temporally Qualified Assertion, TQA)是以对象 $a_1,a_2,\cdots,a_k$ 逐个分别替换 $l_1,l_2,\cdots,l_k$ 后的时态限定表达式。

定义 4 给定一个时态限定命题集合 $TQAs$ 和一个时态变量上的析取时态约束集合 D,一个时态限定命题 $P=\langle t_{p1},p,t_{p2}\rangle$关于$\langle TQAs,D\rangle$是不安全的,如果以下条件成立:

$(\neg\exists\langle t_x,p,t_y\rangle\in TQAs\ s.t.\ t_x\leqslant t_{p1}\wedge t_{p2}\leqslant t_y\in D)\vee(\exists\langle t_x,\neg p,t_y\rangle\in TQAs\ s.t.\ t_y\leqslant t_{p1}\vee t_{p2}\leqslant t_x\notin D)$

$\langle t_s,p,t_e\rangle$的语义是 p 的最小成立区间为$[t_s,t_e)$,在此区间之外仍然按照 STRIPS 规划中的惯性假设:超出该区间仍旧成立,直到由某个动作或事件将其真值逆转;对于某些永恒成立的命题,其成立区间隐含定义为$[0,\infty)$。直观地说,不安全 TQA 是由这两类 TQA 组成的:一类是在 TQA 库中未能得到"支持"的 TQA(不存在针对同样的 p 命题但其成立区间包含该 TQA 成立区间的 TQA,称为开放目标);另一类是可能与 TQA 库中的其它 TQA 存在不一致性而未得到显式解决的 TQA(满足条件$\exists\langle t_x,\neg p,t_y\rangle\in TQAs\ t_y\leqslant t_{p1}\vee t_{p2}\leqslant t_x\notin D$,称为潜在冲突)。这两类 TQA 与部分序规划中的规划"缺陷"类似,但在 LP-TPOP 中并不将这两类 TQA 区别看待,在规划求精的过程中统一求解。在 LP-TPOP 中这两类不安全 TQA 都是需要解决的对象。

定义 5 一个时态限定操作符(Temporally Qualified Operator, TQO)是一个六元组 $o=\langle name,precons,effs,ta,dur,D\rangle$,其中:$name$ 是形如 $o(l_1,l_2,\cdots,l_k)$($l_1,l_2,\cdots,l_k$ 是对象变量符号)的表达式;$precons$ 和 $effs$ 是时态限定命题集合,分别表示动作执行的前提条件和效果;$ts\in\mathbf{Z}$ 是时态变量,$dur\in\mathbf{Z}$ 是一个整数常量,表示该动作的持续时间,D 是定义在 $precons$ 和 $effs$ 中的时态限定命题的时态变量上的析取时态约束集合(称为操作符公理)。给定时态限定操作符 o,时态动作(Temporal Action, TA)$o(q_1,q_2,\cdots,q_k)$是将 o 中的所有参数 $l_1,l_2,\cdots,l_k$ 分别用 $q_1,q_2,\cdots,q_k$ 替换后的操作符[1)]。

定义 6 给定一个不安全的 TQA $R=\langle t_s,p,t_e\rangle$,它的一个安全解(Safe Resolution)是一个二元组:$\langle a,\{t_x\leqslant t_s,t_e\leqslant t_y\}\rangle(\langle t_x,p,t_y\rangle\in effs(a))$($a$ 是个时态动作,此时也称 R 是开放目标),或者$\langle NIL,\{t_y\leqslant t_{p1}\vee t_{p2}\leqslant t_x\}\rangle$(也称 R 是潜在冲突)。

从定义 6 可见,每个安全解均与一个时态约束有关,而这些约束的集合便构成一个析取时态问题[13]。此约束集合的一致性可通过调用析取时态问题求解器进行计算。

定义 7 一个规划领域 Σ 是一个四元组$\langle OBJ,A_{\mathrm{I}},A_{\mathrm{II}},ACT\rangle$,其中:$OBJ=\{obj_1,obj_2,\cdots,obj_k\}$是 k 个对象集合;$A_{\mathrm{I}}=\{fp_1,fp_2,\cdots,fp_n\}$是可变原子命题(真值可能随时间的变化而改变)集合;$A_{\mathrm{II}}=\{rp_1,rp_2,\cdots,rp_e\}$是固有原子命题(真值是固定不变的);$ACT=\{o_1,o_2,\cdots,o_m\}$是时态限定操作集合。

规划领域是对规划系统运行的动态环境的抽象描述。规划领域包括描述系统中关系和性质的原子命题(包括可变命题和固有命题),以及所有可以使用的时态动作。

定义 8 给定规划领域 Σ,Σ的一个规划问题是一个三元组$\langle\Sigma,S,G\rangle$,其中 S 是初始命题集合,G 是目标命题集合。

定义 9 一个部分规划是一个四元组$\langle USF,Acts,Have,D\rangle$,其中:$USF$ 是关于$\langle Have,D\rangle$不安全的 TQA 集合;$Acts$

1) 本文规定:规划算法使用的动作是完全实例化的,不考虑使用部分实例化动作。

是时态动作集合；$Have=\{\langle t_s,p,t_e\rangle \mid \forall a\in Acts\langle t_s,p,t_e\rangle\in effs(a)\}$是所有 $Acts$ 中的时态动作的效果 TQA 所组成的集合；D 是时态约束集合：

$$D=\{t_{s1}\leqslant t_{s2},t_{e2}\leqslant t_{e1}\mid \forall a\in Acts,\langle t_{s2},p,t_{e2}\rangle\in precons(a),\exists a'\in Acts(a'\neq a),\langle t_{s1},p,t_{s2}\rangle\in effs(a')\}\cup\{t_{e1}\leqslant t_{s2}\vee t_{e2}\leqslant t_{s1}\mid \forall a\in Acts,\langle t_{s2},p,t_{e2}\rangle\in effs(a),\forall a'\in Acts(a'\neq a),\langle t_{s1},\neg p,t_{s2}\rangle\in effs(a')\}$$

定义 10 给定规划问题$\langle\Sigma,S,G\rangle$，它的一个有效规划 SP 是一个部分规划$\langle USF,Acts\cup\{a_0,a_g\},Have,D\rangle$，使得 $USF=\emptyset$，且 D 是一致的；其中，a_0 和 a_g 是两个哑动作(分别称为初始哑动作和目标哑动作)：

$a_0=\langle a_0,precons=\emptyset,effs=\{\langle t_0,p_i,t_{0i}\rangle\mid p_i\in S\},t_0\geqslant 0,D=\{t_0\leqslant t_{0i}\mid\langle t_0,p_i,t_{0i}\rangle\in effs\}\rangle$

$a_g=\langle a_g,precons=\{\langle t_g,q_i,+\infty\rangle\mid q_i\in G\},effs=\emptyset,t_g\geqslant 0,D=\emptyset\rangle$[2)]

3 LP-TPOP 算法

3.1 算法描述

图 1—图 4 给出了 LP-TPOP 算法及其子过程的描述。

```
procedure LP-TPOP
Input: partial plan pl=<NSF, A, Have, D>;
Output: a solution plan if solvable, ∅ otherwise;
1. if NSF=∅ return <NSF, A, Have, D>;
2. n=NSF_select(NSF);
3. for each resolver in RESOLVERS(n)
4.    c=RESOLVER_select(RESOLVERS(n), n);
5.    if applyResolver(pl, n, c) then
6.    res=LP-TPOP(<NSF, A, Have, D>);
7.    if res!=∅ then return res;
8.    recoverState();
9. return ∅;
```

图 1 LP-TPOP 算法

LP-TPOP 算法是典型的 DCSP 回溯求解算法：该算法每次选择一个不安全的 TQA(相当于 CSP 中的变量)(第 2 行)，从其安全解(RESOLVERS(n))中选择一个“安全解”(相当于 CSP 中的值)为当前变量赋值(第 4 行)。若应用该安全解不会导致任何不一致性(第 5 行)，则进行下一轮迭代(第 6 行)。若该安全解将导致不一致的时态约束，或者下一轮迭代到达死点，将部分规划解恢复到应用该安全解之前的状态(第 8 行的 recoverState()函数)，继续尝试选择其它的安全解。

图 2 中的 applyResolver 过程展示了安全解应用的具体过程。它首先检查不安全 TQA 及其安全解的类型。若 n 是个开放目标且其安全解约束集合来自新动作(不是 A 中已有的动作)的效果(第 1 行)，则首先将该动作加入到已有动作集合 A 中(第 2 行)，将该安全解约束集合加入到当前时态约束集合 D 中(第 3 行)；另外，还需要将该动作的所有前件 TQA 作为新的不安全 TQA 加入到 NSF 集合中(即将该新的变量置为 active，第 5 行)。加入新的不安全 TQA 的同时，生成它们的安全解(变量的值)集合(第 6 行的 generate_resolvers 函数)。此外，对于该动作的所有效果 TQA，将其作为已有 TQA 集合加入到 $Have$ 集合中(第 8 行)，同时其对于$\langle A,Have\rangle$可能是新的不安全 TQA(潜在冲突)，生成该不安全 TQA 及其安全解集合(第 9 行)。若 n 是个开放目标且其安全解约束集合来自旧动作(A 集合中已有的动作)(第 10 行)，则将该安全解约束集合加入到集合 D 即可(第 11 行)。否则，若 n 是个潜在冲突，则也将其安全解约束集合加入到集合 D 中(第 12 行)。最后，调用 DTP 求解系统 DTP_solve[14]，对更新的约束集合 D 进行求解并将其结果返回(第 13 行)。generate_resolvers 函数分别从规划问题的所有时态动作(新动作，见图 2 第 2 行)和当前部分规划的已有动作(见图 3 第 5 行)中生成不安全 TQA 的安全解。而 NSF_generate_resolvers函数从当前部分规划的已有 TQA 库($Have$)中寻找与之冲突的 TQA，找到则将 r 加入到新的不安全 TQA 库(NSF)中，并生成其安全解集合 RESOLVERS(r)。这里需要注意的是，对于潜在冲突，只需要考虑从已有动作的效果中寻找冲突和安全解，不需要考虑动作前件中与之冲突的 TQA，这是因为动作前件 TQA 本身也是个不安全 TQA，最终也会有其安全解，该安全解必然与当前的潜在冲突之间形成冲突，从而会根据 NSF_generate_resolvers 来形成新的安全解。

```
function applyResolver
Input: partial plan pl=<NSF, A, Have, D>, n-unsafe TQA, c-a
       resolver of n;
Output: true if succeed, false otherwise;
1. if c=<a,{t_s ≤ t_a ∧ t_y ≤ t_e}>(n=<t_a,p,t_e>) and a∉A then
   //n is an open-goal, and a is a new action;
2.     A=A∪{a};
3.     D=D∪{t_s ≤ t_a, t_y ≤ t_e}
4.     for each TQA r in precons(a)
5.        NSF=NSF∪{r}; // add new unsafe TQA;
6.        generate_resolvers(r);
7.     for each TQA r=<t_s,p,t_y>∈effs(a)
8.        Have=Have∪{r};
9.        USF_generate_resolvers(NSF, Have, r);
           // generate possible unsafe TQA and its resolvers;
10.  else if c=<a,{t_s ≤ t_a ∧ t_y ≤ t_e}>(n=<t_a,p,t_e>) and a∈A then
     //n is an open-goal, but a is an action already in A;
11.     D=D∪{t_s ≤ t_a, t_y ≤ t_e};
12.  else if c=<NIL,{t_s ≤ t_a ∨ t_y ≤ t_a}>(<t_a,¬p,t_e>∈effs(a'),a'∈A) then
     // n is a potential conflict;
13.     D=D∪{t_s ≤ t_y ∨ t_y ≤ t_e};
14.  return DTP_solve(D);
```

图 2 applyResolver 子过程

```
function generate_resolvers
Input: r=<t_s,p,t_e>-an open-goal;
Output: RESOLVERS(r)-set of resolvers for r;
1. RESOLVERS(r)=∅;
2. for each action a_i in all temporal actions Acts={a_1,a_2,...,a_k}
3. if ∃<t_s,p,t_y>∈effs(a_i) then
4.        RESOLVERS(r)=RESOLVERS(r)∪{<a_i,{t_s ≤ t_a ∧ t_y ≤ t_y}>};
5. for each action a in A
6. if ∃<t_s,p,t_y>∈effs(a) then
7.        RESOLVERS(r)=RESOLVERS(r)∪{<a,{t_s ≤ t_a ∧ t_y ≤ t_y}>};
```

图 3 generate_resolver 子过程

2) 注意：本文定义的有效规划并不给出动作的确切时间调度，而是给出保证可调度的动作集合。

```
function NSF_generate_resolvers
Input: original NSF, Have-existing TQAs, r=<t_x,p,t_y>-a
       potential conflict;
Output: updated NSF, and RESOLVERS(r) for new NSF r;
1. RESOLVERS(r)=∅;
2. for each TQA w=<t_s,¬p,t_e>(∈effs(a))∈Have
3.    RESOLVERS(r)=RESOLVERS(r)∪{<a,{t_e≤t_x∨t_y≤t_s}>};
4. if RESOLVERS(r)≠∅ then NSF=NSF∪{r};
5. return NSF, RESOLVERS(r);
```

图 4　NSF_generate_resolver 子过程

3.2　算法实例演示

假设有一个正在太空飞行的飞行器。在飞行过程中，飞行器与地面的通信，以及进行科学观察必须在某些“时间窗”内进行。由于飞行器需要进行的科学观察比较多而时间窗是有限的，因此必须对不同的科学观察设定一个优先次序。进行每个科学观察之前飞行器需要转向(Turn)并校准来对准观察目标(Calibrate)，然后进行拍照(TakeImage)。飞行器的转向操作耗时较长，因此需对各种观察进行必要的规划和调度。

根据第 2 节相关概念的定义，该规划领域 Σ 的定义如图 5 所示。给定初始状态 $S=\{Pointing(Earth), On(Camera2), Calibration(T17)\}$，目标状态 $G=\{Image(A37)\}$ 的规划问题，图 6 给出了一个有效规划。图 7 是该有效规划的图示。图 7 中实线方框表示一个时态动作，而虚线方框表示一个时态限定命题 TQA。其中，实线箭头表示的是命题对动作的“支持”关系，或者动作对命题的“派生”关系，而虚线箭头表示某动作的效果 TQA 与另一个动作的效果 TQA 之间存在着潜在冲突。图 7 中的时间区间$[t_x, t_y)$表示 TQA 成立的区间(对虚线方框中的 TQA，其成立时间区间的端点用上下两个时间变量来表示，如对于 $calibrated(Camera2)$ 成立的区间为$[t_{13}, t_{14})$)，标于动作之上的时间点 t_x 表示该动作的开始执行时间。

$\Sigma=<OBJ, A_f, A_{fl}, Acts>$

$OBJ=\{Earth, T17, A37, Camera1, Camera2\}$

$A_f=\emptyset$

$A_{fl}=\{Pointing(?target), On(?instrument), Calibration-Target(?target), calibrated(?instrument), Image(?target)\}$

$Acts=\{Turn, Calibrate, TakeImage\}$

$Turn=<Turn(?target1,?target2), precons, effs, t_a, dur=30, D>$

$precons=\{<t_1, Pointing(?target1), t_2>\}$

$effs=\{<t_5, \neg Pointing(?target1), t_6>, <t_7, Pointing(?target2), t_8>\}$

$D=\{t_a\le t_2, t_a=t_5, t_a+3=t_7\}\cup\{t_i\le t_{i+1}\mid i=1,3,5,7\}$

$Calibrate=<Calibrate(?instrument,?target), precons, effs, t_b, dur=10, D>$

$precons=\{<t_9, On(?instrument), t_{10}>, <t_{11}, Calibration-Target(?target), t_{12}>, <t_{13}, Pointing(?target), t_{14}>\}$

$effs=\{<t_{15}, \neg On(?instrument), t_{16}>, <t_{17}, calibrated(?instrument), t_{18}>\}$

$D=\{t_b+10\le t_{10}, t_9\le t_b, t_b\le t_{12}, t_b+10\le t_{14}, t_{13}\le t_b, t_b+10=t_{15}, t_b+10=t_{17}\}\cup\{t_i\le t_{i+1}\mid i=9,11,13,15,17\}$

$TakeImage=<TakeImage(?target,?instrument), precons, effs, t_c, dur=5, D>$

$precons=\{<t_{19}, calibrated(?instrument), t_{20}>, <t_{21}, pointing(?target), t_{22}>\}$

$effs=\{<t_{23}, Image(?target), t_{24}>\}$

$D=\{t_{19}\le t_c, t_c+5\le t_{20}, t_{21}\le t_c, t_c+5\le t_{22}, t_{23}=t_c\}\cup\{t_i\le t_{i+1}\mid i=19,21,23\}$

图 5　太空飞行器领域描述

$pp=<\emptyset, A, Have, D>$

$A=\{a_0, Calibrate(Camera2,T17), Turn(Earth,T17), Turn(T17,T37), TakeImage(Camera2,T37)\}$

$Have=\{<t_0, On(Camera2), t_1>, <t_0, Calibration-Target(T17), t_2>, <t_0, Pointing(Earth), t_3>, <t_{28}, Pointing(T17), t_{29}>, <t_{33}, \neg Pointing(Earth), t_{34}>, <t_{21}, Pointing(T37), t_{22}>, <t_{24}, \neg Pointing(T17), t_{25}>, <t_{11}, \neg On(Camera2), t_{12}>, <t_{13}, calibrated(Camera2), t_{14}>, <t_{20}, Image(A37), \infty>, <t_g, Image(A37), \infty>\}$

$D=\{t_0\le t_3, t_6\le t_1, t_0\le t_7, t_8\le t_2, t_0\le t_{31}, t_{32}\le t_3, t_{28}\le t_{26}, t_{27}\le t_{29}, t_{13}\le t_{16}, t_{17}\le t_{14}, t_{21}\le t_{18}, t_{19}\le t_{22}, t_{20}\le t_g, t_1\le t_{11}\vee t_{12}\le t_0, t_{25}\le t_{28}\vee t_{29}\le t_{24}, t_{34}\le t_0\vee t_3\le t_{33}\}\cup\{t_{32}=t_{30}, t_{33}=t_{30}+3=t_{28}\}\cup\{t_{27}=t_{23}, t_{24}=t_{23}+3=t_{21}\}\cup\{t_5\le t_4, t_4+10\le t_6, t_9\le t_4, t_4+10\le t_{10}, t_2=t_7, t_4+10=t_{11}, t_4+10=t_{13}\}\cup\{t_{16}\le t_{15}, t_{15}+5\le t_{17}, t_{18}\le t_{15}, t_{15}+5\le t_{19}, t_{15}+15=t_{20}\}\cup\{t_0\le t_1, t_0\le t_2, t_0\le t_3, t_{31}\le t_{32}, t_{33}\le t_{34}, t_{28}\le t_{29}, t_5\le t_6, t_7\le t_8, t_9\le t_{10}, t_{11}\le t_{12}, t_{13}\le t_{14}, t_{26}\le t_{27}, t_{24}\le t_{25}, t_{21}\le t_{22}, t_{16}\le t_{17}, t_{18}\le t_{19}\}$

图 6　太空飞行器规划问题的有效规划

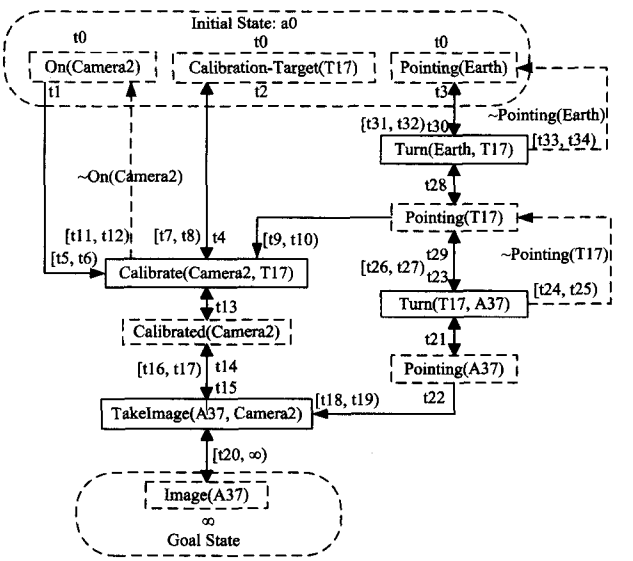

图 7　规划图解

3.3　算法可靠性和完备性

性质 1(LP-TPOP 的可靠性)　LP-TPOP 是可靠的，即由 LP-TPOP 计算得到的规划解是有效的规划解。

证明(简要)：给定规划问题 $P=\langle\Sigma, S, G\rangle$，设 LP-TPOP 返回一个解规划$\langle NSF, Acts, Have, D\rangle$。根据 LP-TPOP 算法定义，$NSF=\emptyset$ 必定成立(LP-TPOP 算法第 1 行)。D 也一定成立，否则在最后一次应用安全解(运行 applyResolver 函数)过程中，最后调用 DTP 求解算法 DTP_solve 函数(applyResolver 函数第 14 行)必定返回 false，从而导致该安全解被抛弃(LP-TPOP 算法第 8 行)而尝试其它安全解。证毕。

性质 2(LP-TPOP 的完备性)　LP-TPOP 是完备的，即如果规划问题存在一个有效规划解，则 LP-TPOP 必能找到这样一个解规划。

证明(简要)：给定规划问题 $P=\langle\Sigma, S, G\rangle$，设 P 存在一个有效规划解$\langle NSF, Acts, Have, D\rangle$($NSF=\emptyset$，$Acts=\{a_0, a_1, a_2, \cdots, a_l\}$)(不失一般性，设 $Acts$ 中时态动作的加入顺序正是 $a_1, a_2, \cdots, a_l$，且不存在 $Acts$ 的子集构成有效规划解)。根据系统搜索算法的完备性，当已有部分解规划$\langle NSF', \{a_0,$

$a_1, a_2, \cdots, a_k, a_g\}, Have', D'\rangle (NSF' \neq \emptyset, k<l)$的情况下，迭代运行 LP-TPOP 时，不安全 TQA 集合为$\{nsf_1, nsf_2, \cdots, nsf_i, nsf_{i+1}, \cdots, nsf_h\}$，必存在 nsf_i 是开放目标且有 $a_{k+1} \in$ RESOLVERS(nsf)。根据假设下一个最早选择的不安全 TQA 为 nsf_i。同时，若选择 $a'(\neq a_k)$，在 LP-TPOP 算法下的 for 循环(第 3 行)中，继续往下迭代只有两种结果：或者找到一个解退出(第 7 行)，或者到达死点尝试下一个安全解。若第一种情况，证明结束。否则，当选择 a_k 作为安全解，得到的部分解规划必然是一致的(LP-TPOP 第 5 行调用 applyResolver 函数返回 true)。如此迭代下去，必能最终找到有效规划解$\{a_0, a_1, a_2, \cdots, a_l\}$。

结束语 本文基于动态约束满足框架设计了一个基于约束区间时态规划算法 LP-TPOP，与同样基于 DCSP 的 Smith 的 CBI 算法和 Europa/Europa2 规划算法相比，LP-TPOP 具有以下特点：

(1)CBI 算法使用 STP 作为时态推理模型，而 LP-TPOP 使用 DTP 作为时态推理模型。这使得 LP-TPOP 在规划和调度模块上具有更合理的计算量分布[21]。

(2)Europa/Europa2 使用以状态变量为中心的表达语言，而 LP-TPOP 的 CBI 规划是以动作为中心。对一个实际现实世界规划问题，LP-TPOP 的变量数目比 Europa/Europa2 少得多；虽然同样基于 DCSP，但是对于生成的 DCSP 问题规模，LP-TPOP 比 Europa/Europa2 要小些。

虽然 LP-TPOP 同样具有多数领域无关的通用规划算法所具备的可靠性和完备性，但是基于此算法的实用规划系统的实现和实验验证同样不是一个容易完成的工作，因为除核心算法外，一个 CBI 规划系统还涉及到 CBI 形式语言定义和解析、DTP 求解器的使用、CBI 测试实例的构建(CBI 规划的实验例子仍然是缺乏的)等问题。这些作为后续工作将会在未来研究中完善。

参 考 文 献

[1] Green C. Application of theorem proving to problem solving [C]// Preceedings of IJCAI. Washington DC, USA, 1969: 741-747

[2] Traverso P, Ghallab M, Nau D. Automated planning: theory and practice [M]. San Francisco: Morgan Kaufmann Publishers, 2004

[3] Smith D, Weld D S. Temporal planning with mutual exclusion reasoning [C]// Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. Stockholm, Sweden, July-August, 1999: 326-337

[4] Fox M, Long D. PDDL2.1: An extension to PDDL for expressing temporal planning domains [J]. In Journal of artificial intelligence research, 2003, 20: 61-124

[5] Fikes R E, Nilsson N J. STRIPS: A new approach to the application of theorem proving to problem solving [J]. Artificial Intelligence, 1971, 2(3/4): 189-208

[6] McDermott D. PDDL: the Planning Domain Definition Language [R]. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998

[7] Bacchus F. The power of modeling-a response to PDDL2.1[J]. J. Artificial Intelligence. Res. (JAIR), 2003, 20: 125-132

[8] Cushing W, Mausam, Kambhampati S, et al. When is Temporal Planning Really Temporal? [C]// Proceedings of IJCAI 2007. Hyderabad, India, January 2007: 1852-1859

[9] Smith D E. The case for durative actions: A Commentary on PDDL2.1[J]. Journal of Artificial Intelligence Research, 2003, 20: 149-154

[10] Smith D E, Frank J, Jonsson A, et al. Bridging the Gap between Planning and Scheduling [J]. Knowledge Engineering Review, 2000, 15(1): 47-83

[11] Frank J, Johnson A. Constraint based attribute and interval planning[J]. Journal of Constraints, Special Issue on Planning, 2003, 8(4): 339-364

[12] Mittal S, Falkenhainer B. Dynamic constraint satisfaction problems[C]// Proceedings AAAI Conference. Boston, Massachusetts, July-August 1990: 25-32

[13] Stergiou K, Koubarakis M. Backtracking Algorithms for Disjunctions of Temporal Constraints[C]// Proceedings of the 15th National Conference on Artificial Intelligence(AAAI-98). Madison, Wisconsin, July 1998: 248-253

[14] Liu Yue-chang, Qian Hong, Jiang Yun-fei. Topology-based Variable Ordering Strategy for Solving Disjunctive Temporal Problems [C]// TIME. 2008: 129-136

[15] Bernardini S, Smith D E. Translating PDDL2.2 into a Constraint-based Variable/Value Language [C]// Proceedings of the Workshop on Knowledge Engineering for Planning and Scheduling(KEPS). 18th International Conference on Automated Planning and Scheduling(ICAPS-2008). 2008

[16] Bernardini S. Constraint-based Temporal Planning: Issues in Domain Modelling and Search Control [D]. Italy: The University of Trento, 2008

[17] Bernardini S, Smith D E. Towards Search Control via Dependency Graphs in EUROPA2[C]// Proceedings of the 2nd Workshop on Heuristics for Domain-independent Planning, 19th International Conference on Automated Planning and Scheduling (ICAPS-2009). 2009

[18] Bernardini S, Smith D E. Developing Domain-Independent Search Control for EUROPA2 [C]// Proceedings of the Workshop on Heuristics for Domain-independent Planning: Progress, Ideas, Limitations, Challenges, 17th International Conference on Automated Planning and Scheduling(ICAPS-2007). 2007

[19] 刘越畅，姜云飞，钱红. 基于问题结构的启发式策略在析取时态问题求解中的应用[J]. 计算机研究与发展，2008，45(11)：1840-1849

[20] 刘越畅. 基于约束的时态推理和时态规划[D]. 广州：中山大学，2008

[21] Nguyen Xuan-long, Kambhampati S. Reviving partial order planning [C]// Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI). Seattle, USA, 2001: 459-464