基于强化学习的业务流程中的柔性约束研究

韩道军[1,2] 夏兰亭[1] 卓汉逵[1] 李 磊[1]

(中山大学软件研究所 广州 510275)[1] (河南大学数据与知识工程研究所 开封 475004)[2]

摘 要 软件业务流程需要较高的灵活性和适应性。目前,众多的方法集中在流程建模阶段,通过一些方法提高流程的灵活性,却忽略了流程的运行阶段。由于外界环境的变化具有动态性,在建模阶段不易对其描述。由于外界环境的变化能够在运行阶段体现出来,分析流程的运行阶段是必要的。以流程中的约束为研究内容,通过对流程运行阶段的数据进行分析,提出一种基于强化学习的柔性约束模型,以提高流程的适应性。同时,将该算法应用于一类以用户为中心的复杂信息系统,实例分析表明算法是实用和有效的。

关键词 强化学习,柔性约束,业务流程

中图法分类号 TP301 **文献标识码** A

Research on Flexible Constraint of Business Process Based on Reinforcement Learning

HAN Dao-jun[1,2] XIA Lan-ting[1] ZHUO Han-kui[1] LI Lei[1]

(Software Research Institute, Sun Yat-Sen University, Guangzhou 510275, China)[1]

(Institute of Data and Knowledge Engineering, Henan University, Kaifeng 475004, China)[2]

Abstract Software business process needs higher flexibility and adaptability. Traditionally, various methods are developed to improve the flexibility of process during the modeling period, and the running period is omitted. It is hard to describe the environment model for its dynamic, thus, environment can't be controlled during the modeling period. Because the change of environment can be explained by the data of running period, analyzing the treated data of running period of process is necessary. We built a learning algorithm based on reinforcement learning, a well-known learning technique, to learn flexible constraints from the treated data of running period, and the adaptability of process is improved. We will evaluate our learning algorithm in a user-centred complex information system, and show that our algorithm is effective and efficient.

Keywords Reinforcement learning, Flexible constraint, Business process

软件业务流程(Process)不同于一般意义的工业过程或稳定的实际业务流程,其要求更高的灵活性和适应性[1]。对于业务流程的研究,众多学者提出了一系列的方法及工具[2-8]。其中,文献[1]提出了一种基于 Agent 的软件流程建模方法,用 Agent 封装软件流程中涉及的个体,Agent 通过对环境进行感知,动态建立针对给定软件项目的软件流程模型,并在执行过程中不断更新自身知识和能力以动态适应环境的变化。文献[6]提出一种通过知件模型与用户进行交互的方法,希望用户自己来定义、设计、开发、维护和修改软件。文献[7]对一些业务流程的建模方法进行分析比较,并介绍了一种多维度的集成化软件流程建模方法。文献[8]将面向服务的软件应用需求转化为应用的服务模型,并充分体现了广泛复用、灵活匹配、松散耦合的 SOA 理念。以上方法取得了很好的效果,但是仍然存在灵活性和适应性不够的问题,其根本原因为:在动态变化的环境中,环境的变化具有未知性,则在业务流程的设计阶段很难预定义或预测环境的变化,无法为环境模型定义,从而无法对环境进行建模和设计约束。

对于软件业务流程的适应性,目前的研究主要通过提高软件的灵活性来增加系统的适应性。根据软件的生命周期可知,这种方式属于在建模阶段对业务流程的灵活性进行增强。在软件的运行阶段,同样需要引起我们的注意。这是因为在软件的运行阶段,处理的是具体数据,而数据中隐藏着丰富的知识,能够反映出环境的变化情况。自然,我们可以对运行阶段中业务流程处理的数据进行分析,并将得到的结果反馈给系统,使系统能够自动调整相关业务流程的约束信息,以增加系统适应性。

以业务流程为主要管理对象的系统通常称为复杂信息系统(Complex Information System, CIS)[9,10]。在各类 CIS 中,存在一类重要的 CIS,本文称之为以用户为中心的 CIS(User-Centred Complex Information System, UCCIS),其特征为:业务流程的每一步推进均需要用户的参与,用户的决策对于业务流程的推进具有决定性作用。常见的有:行政审批系统、企业信息系统等。在审批系统中,每个待办事项的决定权都由相应的用户负责,系统仅仅负责将审批申请表根据满足的条

到稿日期:2010-04-16 返修日期:2010-07-27 本文受国家自然科学基金(60773201)资助。

韩道军(1979—),男,博士生,讲师,主要研究方向为机器学习、形式概念分析、软件工程,E-mail:hdj@henu.edu.cn;**夏兰亭**(1982—),男,博士生,主要研究方向为软件工程、数据库与知识库;**卓汉逵**(1982—),男,博士,主要研究方向为智能规划、软件工程;**李 磊**(1951—),男,教授,博士生导师,主要研究方向为数据库与知识库。

件流转至对应的状态。在 UCCIS 中，用户希望系统具有一定的柔性，使得预先设定的流程推进的约束条件具有一定的灵活性，能够适应外界环境（各类输入）的变化。

本文提出一种具有自适应特征的柔性约束模型应用于 UCCIS，并用人工智能的一个核心研究领域——机器学习中的强化学习方法实现。柔性约束模型首先对约束信息中的属性进行量化、加权、度量（计算表单与约束信息之间的距离），然后采取相应的操作。同时从 UCCIS 的系统日志中找出学习样本，从样本中学习出约束信息中阈值的变化量，从而可以使 UCCIS 的约束信息由“硬性”变为“柔性”，即更加灵活，以增强环境适应能力并降低维护代价。

本文第 1 节介绍相关背景知识及提出柔性约束模型；第 2 节就如何应用机器学习的方法实现具有自适应特征的柔性约束模型给出了解释；第 3 节通过实验说明该模型的优越性和先进性；最后给出本文的总结和进一步的工作。

1 相关知识及模型介绍

1.1 业务流程介绍

常见的业务流程开发方式有基于工作流和基于 Agent 的开发方式[11-13]。本文选用文献[11]中的 SPDM 模型，该模型使用术语“表单”来描述应用层面的研究对象，其理由如下：UCCIS 与其他系统有着明显的区别，在这些应用的日常业务操作中，通常以一个或少量几个关键表单（模式）为主，它贯穿该项业务处理的始终。以下是相关的定义。

定义 1（表单） 表单（Form）是机构中的业务表格（如申请表）、单据、任务说明、凭证、证书等的统称，它作为业务数据的载体而存在，也是业务数据的模式，相当于数据模型中的模型（Model）。一个表单中的项目（Item）称为表单项，其取值称为表单项值。

定义 2（业务流程） 业务流程（Process）以表单为研究对象，描述某一表单在其生命周期中经历各种业务操作的状态变化过程。一个业务流程的执行过程是：表单从开始状态出发，经过若干个业务操作到达其终止状态。

定义 3（步） 业务流程中的一步（Step）是指从状态 $State_i$ 到 $State_j$ 的一次转换，其中这两个状态之间必须有一条有向弧直接相连。一般地，一个流程包含若干步。

从上述定义可以看到，流程的推进实际上是状态不断改变的一个过程。明显地，在一个流程的推进过程中，每一步转换都不是没有条件的。为了描述这些前提条件，我们用一个具有较强表达能力的逻辑表达式来丰富这一约束条件。我们称这个表达式为约束公式（Constraint Formulas，CF）。约束公式可以通过从表单中抽取出相应的属性来构造，并参考 ABAC 中 AAR 的描述方式[14]，形式化地表示为 $att_1 = constant_1 \wedge att_2 > constant_2 \wedge \cdots \wedge att_m = constant_m$。

图 1 为一个基于表单状态变迁的业务流程实例描述（审批流程），每个节点代表表单的一个状态，且图中的每一个状态均可以进入异常终止状态。图 1 中的表单状态转换的条件（约束）可以标注在有向弧上。在系统的运行过程中，除非用户手工修改该约束信息，否则不会发生变化。并且，可以看出图 1 中的每个状态都需要有用户的参与，用户参与在整个流程的推进过程中具有决定性作用。

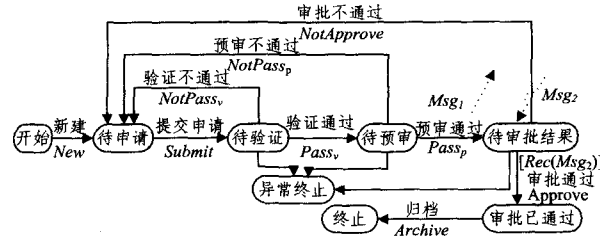

图 1 基于状态的审批流程示意图

1.2 柔性约束模型

现有 UCCIS 中的约束模型不能适应环境的变化，该模型可以称为静态约束模型。本文通过将机器学习的技术引入到 UCCIS 中，使得 UCCIS 中的约束模型具有进化功能，能够随着外部环境的变化而变化。即在约束具有进化功能的 UCCIS 中，约束信息描述不再认为是一种绝对的约束，而是具有一定的弹性。与常规的设定固定阈值 w（表示系统可以接受的信息误差度量，且每条约束信息相关的阈值不同）方式不同，柔性约束中的阈值 w 不再是一成不变的，其能够根据系统中处理过的表单的信息和状态变迁情况动态赋值。本文称这种引入机器学习功能的约束模型为柔性约束模型，该模型的特点为约束信息的阈值 w 能够自动变化。

下面以业务流程中的一步来说明柔性约束模型。在 UCCIS 中，传统的约束模型如图 2 所示。当表单处于状态 S_i 时，收集表单的属性及取值情况，在约束信息库中找出匹配的约束信息，如果与一条约束信息 C_i 匹配成功，则跳转至 C_i 所能到达的状态 S_{i+1}；否则，返回（不做任何操作）。这里的约束信息库是指应用系统中所有与表单状态转换有关的约束构成的集合。

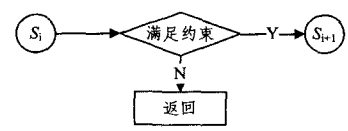

图 2 静态约束信息示意图

具有自适应特征的柔性约束模型如图 3 所示。从图 3 中可以看出，若表单 F_i 在 S_i 时的属性及取值（ATT）不满足现有的任何一条约束信息，则此时将 ATT 信息传送至柔性约束适配器（Flexible Constraint Adapter，FCA），FCA 根据系统的阈值 w 设置情况及现有约束信息，经过计算，产生一条输出信息，跳转至 S_{i+1} 或返回。

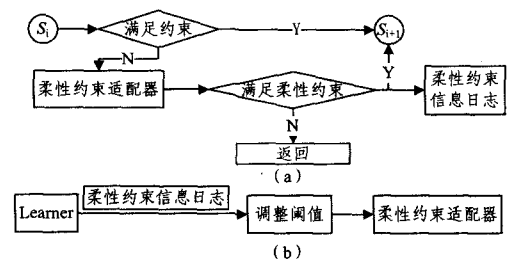

图 3 柔性约束模型示意图

FCA 在运行过程中，将会产生日志信息，该日志仅仅反映柔性约束信息匹配时的计算情况，称为柔性约束信息日志。分析系统的运行日志和柔性约束信息日志，可以对阈值 w 进行调整。阈值 w 能够自动调整，表示该类约束信息模型具有进化功能，能够随着外界的环境变化而智能变化，无需人工修改。

2 柔性约束模型中的关键问题探讨

本节对图3所示的柔性约束模型中的几个关键问题进行讨论，分别对UCCIS中的属性抽取和FCA的工作原理、阈值的动态调整方式进行介绍。

2.1 属性抽取

在SPDM的业务流程图中，表单在当前状态 S_i 包含的属性有：各种数据项属性，如设备名称、购买数量等；访问主体描述信息，如用户ID、角色、信任度等；外部资源属性，如等待消息；动作属性，如提交、审核等。以上各类属性均可以从表单或消息缓冲区中得到，在此不再详细说明。

2.2 柔性约束适配器FCA

FCA的主要功能是处理一些不符合状态跳转约束条件的访问请求。即表单 F_i 在某一状态 S_i 时，在当前约束信息库无法找到其能满足的约束信息，则将 S_i 的属性信息提交至FCA；FCA根据接收的属性信息和约束信息库，通过处理和计算，给出 S_i 应该跳转的下一个状态。当FCA处理完毕后，将会向系统写入一条柔性约束处理日志。

FCA的工作原理如下：

(1)首先在约束信息库中找出所有包含{*InitialState*=S_i}的约束信息集合 *Constraints*；

(2)分别计算 S_i 具有的属性及取值与 *Constraints* 中每个约束的距离度量，记为集合 D；

(3)对于 D 中元素，按照值的从大到小，分别与相应的阈值进行比较，比较 D_i 和 w_i，w_i 为约束信息 $Constraints_i$ 相关的阈值。若 $D_i > w_i$，表示满足当前阈值要求，则选取 $Constraints_i$ 所能到达的状态，即 $Constraints_i$ 中的约束项{*NextState*=S_{i+1}}中的 S_{i+1}，返回 S_{i+1} 并结束。否则处理 D_{i+1}，步骤同上，直至有状态值返回或 D 中元素处理完毕。

在FCA中，核心为通过计算得出最近距离的过程，涉及到属性加权、距离度量设置和数据规范化问题。设定距离的最大值为1，值越大表示越接近现有某条约束信息。按照如下步骤进行说明。

(1)属性排序

一般地，一个属性 a 如果越重要，则会在较多的约束信息中包含对该属性 a 的判定信息。则对于从表单中抽取出来的属性，我们可以在约束信息库中对其进行统计，根据各个属性出现的频率来获取各个属性的重要程度。属性在约束信息库中出现的次数越多，则该属性排序越靠前，将会有较高的权重。即 $Ranking(a) \propto count(a)$，$count(a)$ 表示在约束信息库中属性 a 出现的频率(总次数)。

在将属性排序后，按照序号分别为每个属性赋予一个值，从1开始，到 k 结束，其中 k 为相关属性的总数。

(2)属性加权

获取到属性的序号后，需要对各个属性加权，按照属性的重要程度，分别赋予不同的权重。本文采取的权重分配方式如下：

$$weight(a) = \bar{\omega} + (k/2 - i)\delta$$

式中，$\bar{\omega} = 1/k$，k 为使用的属性总数目，i 为属性 a 的序号，δ 为与 $\bar{\omega}$，k 相关的某个常量，作为调节因子。对于 δ，满足：$0 < \delta < 2\bar{\omega}/k$，使得：$weight(a) \neq weight(b)$，且 $0 < weight(x) < 1$，x 为任意属性。

(3)属性差值计算及数据规范化

根据表单在 S_i 时具有的属性及取值，分别计算与 *Constraints* 中的每条约束信息包含的属性集合中每个属性差值，用 $MinusValue(a)$ 表示，其中 a 为属性。计算出各种属性的属性差值后，要进行规范化处理，将不同的属性差值映射到同一个数据区间，以统一量纲及消除单个属性差值对整体的影响。

参照最小-最大规范化公式：

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A$$

本文设 $new\_\max_A = 1$，$new\_\min_A = 0$，并简化公式为 $v' = \frac{v}{\max_A}$，在后面的计算 $MinusValue(a)$ 时直接应用该公式，即计算出的 $MinusValue(a)$ 即为 v' 的值，在此不再进行详细说明。

在计算MinusValue(a)时，按照属性a的数据类型(数值型和非数值型)不同采用不同的处理方式。

1) 数值型。对于约束信息库 *Constraints* 中的数值型属性约束，分为4种类型：

a)相等，如：(level='3')；

b)大于(大于等于)，如(salary>'3000')；

c)小于(小于等于)，如(salary<'2000')；

d)介于某个范围之间，如(salary>'2000' ∧ salary<'4000')。

在计算 $MinusValue(x)$ 时，x 代表数值型属性，设属性 x 在 S_i 中取值为 v_x，在约束信息中指派的值为 r_x，此处的 r_x，对于a,b,c 3种类型，可以直接从约束信息中获取到；对于 d，则取数值与 v_x 最近的那个值，即

$$r_x = \begin{cases} r_1, & |v_x - r_1| < |v_x - r_2| \\ r_2, & \text{else} \end{cases}$$

式中，r_1，r_2 代表属性 x 取值范围两端的值。

则 $MinusValue(x) = \frac{|v_x - r_x|}{r_{x'}}$

其中，$r_{x'} = \max\{v_x, r_x\}$。

2) 非数值型。对于约束信息库 *Constraints* 中的非数值型属性约束，分为两种类型：

a) 相等，如 color='red'；

b)不等，如 grade>'pass'。

在计算 $MinusValue(x)$ 时，属性 x 为非数值型，属性取值范围 $X = \{v_{x1}, v_{x2}, \cdots, v_{xk}\}$，设属性 x 在 S_i 中取值为 v_{xi}，在约束信息中指派的值为 r_x，分为两种情况分别进行处理：

$$\text{a)}MinusValue(x) = \begin{cases} 1/k, & v_{xi} \neq r_x \\ 0, & \text{else} \end{cases}$$

式中，$k = |X|$。

b)对于不等的情况，需要结合给定的属性取值范围 X 进行处理。在约束信息中设置不等操作，说明该属性的取值是有序的，可比较。可以将这种属性的所有取值按照偏序或全序关系分别赋予不同的数值，然后按照属性取值分配到的数值进行计算。在对 X 中的元素进行赋值时，设 v_x 赋值为 i，r_x 赋值为 j。为简化计算，不再区分 X 中元素的全序和偏序赋值情况，统一令 $k = |X|$。则此时：

$$MinusValue(x) = |i - j|/k$$

在数据计算过程中，需要考虑到两种情况：

a)属性值缺失。对于属性 x 为数值型的，根据系统的表单日志，拟合出一个平均值替代参与 $MinusValue(x)$；属性为

非数值型的，令 $MinusValue(x)=1/k$，其中 k 为属性 x 的所有可能取值的数目。

b)属性缺失。若 *Constraints* 中某条约束信息中需要使用某个属性 x，而在状态 S_i 中不包含属性 x，则令 $MinusValue(x)=1$。

(4)距离度量与次态返回

在距离计算时，常用的有加权欧氏距离、加权曼哈顿距离及闵可夫斯基距离。本文在对属性差值规范化的基础上，采用加权曼哈顿距离计算公式，即：

$$d(i,j)=1-\sum_{k=1}^{m} weight(x_k)\times MinusValue(x_k) \qquad (1)$$

式中，m 为相关属性总数，x_k 表示任一属性，$d(i,j)$ 表示表单在状态 S_i 具有的属性值和某条约束信息的距离度量值。

对于 *Constraints* 中的每条约束信息 r，计算出 r 与表单在状态 S_i 的距离度量值 d_i，得到距离值集合 $D=\{d_1,d_2,\cdots,d_m\}$，$m=|D|$。令 $d_i=\min\{d_1,d_2,\cdots,d_m\}$，若 $\{d_1,d_2,\cdots,d_m\}$ 中的极小值不止一个，则任选一个极小值赋予 d_i。由 d_i 得到对应的约束信息 r_i，根据 r_i 中的 $\{NextState=S_{i+1}\}$ 得出次态为 S_{i+1}，即 FCA 的返回值为状态 S_{i+1}。

按照式(1)进行计算时，由于对于每个属性都进行了加权处理，使得按照同一条约束信息 C_i 进行评判时，不同的表单在某一状态 S 处的度量值的大小反映了信息满足 C_i 的程度，$d(i,j)$ 值越大越接近，较为真实地反映了现实状况。

2.3 学习算法介绍

如柔性约束模型图所示，FCA 返回一个状态后，将会向系统写入一条柔性约束信息日志，作为一种反馈信息。这种日志信息是学习新的阈值的样本空间，需要对其进行标注。在获取到正负样本后，使用 online 学习的一种——强化学习(Reinforcement Learning，RL)算法即可得到新的阈值。下面分别介绍样本的获取和学习的过程。

2.3.1 样本的获取

机器学习离不开样本。从图 3 所示的柔性约束模型中可以看出，FCA 在经过计算得到次态 S_{i+1} 后，将会写入一个柔性约束信息日志。客观上，并不是 FCA 返回的状态 S_{i+1} 都能得到系统的支持，需要对这个过程进行分析处理。若表单从 S_i 经过 FCA 到达 S_{i+1} 后，此时需要对 S_{i+1} 的状态进行评审，即由某个动作 action 将要对处于状态 S_{i+1} 的表单进行操作，其中 action 被看成一种属性。在以 UCCIS 中，动作的施加主体为用户，则可以通过用户的主观判断，由用户给出 S_{i+1} 的次态 S_{i+2}。前文介绍，在基于 SPDM 的流程图中，若 S_{i+2} 为异常结束，则说明此次表单日志是个负样本(E_-)；若 S_{i+2} 是流程图中 S_{i+1} 的正常次态之一，则说明此次表单日志为正样本(E_+)。获取正、负样本后，可以与强化学习中的奖赏函数匹配，设定对应的值，在下文中详细介绍。

2.3.2 基于强化学习的阈值动态设置

在获取到系统的柔性约束日志作为正负样本(E)之后，可以从中学习出新的阈值和属性的权重。考虑到这类系统的特殊性，在系统运行中将会依次产生样本，即不能一次获取全部的样本，本文使用在线学习的一种——强化学习算法处理。

强化学习(Reinforcement Learning)是一种重要的在线学习方式[15-17]。强化学习的基本原理是：如果 Agent 的某个行为策略导致环境对 Agent 正的奖赏(Reward)，则 Agent 以后采取这个行为策略的趋势会加强。其基本模型如图 4 所示。

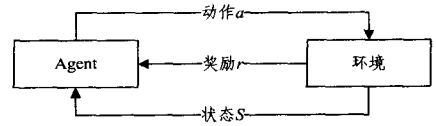

图 4 强化学习基本模型

通常，强化学习问题可以抽象为马尔科夫决定过程(Markov Decision Process，MDP)。MDP 被形式化定义为四元组 (S,A,T,R)，其中 S 指环境的有限状态集合，A 指有限的动作集合，$T:S\times A\to\prod(S)$ 是状态转移函数，$R:S\times A\to R$ 是奖赏值函数。从另一个角度看，强化学习系统主要组成要素有：策略(Policy)、奖赏函数(Reward Function)、值函数(Value Function)以及环境(Environment)的模型(不是必须的)。

在柔性约束模型中，待寻找的策略是表单 F_i 在状态 S 下所应该执行的动作。动作有两种：提交(submit)或返回(cancel)。奖赏函数与表单的状态变迁有关，提交的表单若通过用户的审批(正样本)，则给出奖励(值为正)，否则为惩罚(值为负)。值函数的设计与系统目标有关。引入柔性约束的目的是使系统在现有约束的情况下，提供一些接近现有约束的表单，使得这些表单通过用户审批的可能性较大。假设阈值 w 较大，即与现有约束较接近，则表单提交审批的通过率(用 p 表示)较大，但是不一定能够保证表单审批的数量较大(用 *sum* 表示)；假设阈值 w 较小，即系统中提交审批的表单数量较大，此时审批的表单数量可能会变大，但是，这将使审批通过成功率 p 较小，也即增加了审批用户的代价，使得约束的意义很弱。通过使用强化学习，使得在环境变化的情况下，找到一个动态的阈值 w，使之能够较好地适应环境，p 和 *sum* 的取值尽可能接近期望值。在这两个目标中，p 优先满足。也即，在保证成功率 p 的前提下，尽可能多地得到处理成功的业务 *sum*，也即动态调整阈值 w，使得 p 和 *sum* 的取值尽可能大，其中 p 和 *sum* 的 w 关系分别为：$p\propto w$，$sum\propto(1/w)$。

假定在某一个时间段内，系统需要处理的表单集合为 F_0，其中满足现有约束表单集合为 F_1，满足柔性约束并得到审批的表单集合为 F_2，满足柔性约束但未得到审批的表单集合为 F_3，其他的为 F_4，则 $F_0=F_1\cup F_2\cup F_3\cup F_4$。其中，$F_1$ 部分的表单与柔性约束模型无关，不予讨论。显然，若阈值 $w=0$，则 F_2 均能得到审批，但是此时 F_3 中的表单亦经过审批，约束没有产生任何作用，$p=|F_2|/(|F_2|+|F_3|+|F_4|)$ 为理论最小值；若阈值 $w=1$，则 F_2 和 F_3 均无法进入审批流程，此时体现不出柔性的作用。本文引入强化学习，目的是通过对环境的试探，结合系统返回的审批结果，设置一个随着环境变化的阈值 w，使 p 尽可能接近 1，*sum* 尽可能接近 $|F_2|$，其中 p 为第一目标，*sum* 为第二目标，同时对 p 有约束作用。

结合柔性约束模型，本文假设在线学习中的约束条件“及时反馈”能够满足。即系统提交的表单 F_i 都能够及时地得到审批结果(通过或退回)。对强化学习中的几个要素分别设计如下：

状态 $S=\{S_0,S_1\}$，其中 S_0 为满足柔性约束的表单状态，S_1 为不满足柔性约束的状态。

动作 $A=\{Submit,Cancel\}$。

Policy：$S_0\to Submit$，$S_1\to Cancel$。

Reward Function：

$$Reward(F_i)=\begin{cases}10, & F_i \text{ 为 } E_+\\ -10, & F_i \text{ 为 } E_-\end{cases}$$

其中，数值 10 和－10 无实际量化意义，仅表示正负取值。

与传统强化学习的目标不同，本文的目的是通过学习得到阈值 w，而不是策略。这里的 w 影响状态值，从而间接地影响到了策略。

对于阈值 w 的调整，可以根据表单的状态变化情况以及系统中的历史数据动态调整。在调整阈值时，借鉴 SVM 的分类思想，选取其中的一些状态度量值作为支持向量，分别为 v_+ 和 v_-，使选取的阈值具有较好的分类效果。初始，$v_+=v_-$，$w=(v_++v_-)/2$。对于单个表单的处理步骤如下：

(1)对于表单 F_i，获取 F_i 在状态 S_0 的度量值 $Value(F_i)$。

(2)若 $Value(F_i)>w$，按照 $Policy$，则 FCA 执行动作 $Submit$，将 F_i 提交至下一个状态，转(3)，否则执行动作 Cancel。

(3)若 $Reward(F_i)<0$，$v_+=\max(w)$，$v_-=Value(F_i)$，$w=(v_++v_-)/2$。

显然，按照这种方式，w 的值只能变大，不能变小，这不符合柔性约束模型的要求。为了使阈值能够非单向变化，体现用户的行为，需要使用某种策略对阈值进行向下调整(w 变小)。一种常见的策略是对 FCA 处理过的信息统计分析，从历史数据中找出表单状态信息的变化规律。可以将经过 FCA 处理过的信息分为 3 类：审批通过 F_2、审批未通过 F_3、未获得审批机会 F_4。前两类指表单状态度量值大于阈值 w，后者指表单状态度量值小于阈值 w，未能通过 FCA。记 v_p 为表单状态值度量的平均值，

$$v_p=\frac{\sum Value(F_i)}{|F|}$$

记 l 为柔性日志中两个正样本(奖励为正)之间的距离(初始值为一个随机整数)，则按照逐步逼近的方式对 w 的值进行修改：$w_{t+1}=w_t-\frac{|w_t-v_p|}{l}$，其中 w_t 为当前阈值，w_{t+1} 为处理表单(该表单审批通过或未进入提交审批)之后的阈值。这种方式能够使 w 以一个很慢的方式向较小值方向改变，达到强化学习中 Agent 主动对环境试探的目的。环境的变化即表单的状态度量值总体趋势发生了变化，它按照公式对 w 的值进行调整时，v_p 值在发生变化，总是能够从用户处得到反馈，根据奖赏函数对 w 的值进行调整。则将上述步骤(3)改为：若 $Reward(F_i)<0$，$v_+=\max(w)$，$v_-=Value(F_i)$，$w=(v_++v_-)/2$；否则：$w_{t+1}=w_t-\frac{|w_t-v_p|}{l}$。

2.4 算法描述

表单 F_i 在处于状态 S_i 时，若不符合现有约束信息，则提交至 FCA，FCA 的处理过程(框架式)可以用算法伪码描述如下：

1)对于 F_i 在 S_i 时的信息，取出相应约束信息中的每个属性 a，从约束信息库中统计出 a 出现的频率；

2)根据出现的频率对属性进行排序；

3)对每个属性赋予不同的权重；

4)根据属性权重及对应属性取值情况，计算表单 F_i 与约束信息 C_i 的距离度量 d_i；

5)if $d_i>w_i$，则将表单 F_i 提交至 C_i 所能到达的状态 S_{i+1}；

if $Reward(F_i)>0$，w_i 根据微调因子减小，作为对环境的主动试探操作；

if $Reward(F_i)<0$，w_i 根据 d_i 和取值历史增大，作为对环境变化的反映。

3 实验及分析

本节用高等学校中的国家奖学金的申请和审批流程作为实验场景来说明柔性约束模型的工作方式。由于本文提出的柔性约束模型可以替代以用户为中心的 CIS 中的约束，故不再详细介绍实验环境(系统架构及配置情况等)，仅仅介绍柔性约束的计算过程和为用户带来的便利。

图 5 是一个实际申报的流程。在这个申报流程中，我们可以看出其中的几个特点：(1)约束信息(申请条件)并不是十分严格，与实际情况紧密相关；(2)申请和审批的过程中，用户(审批人)起主要作用。该流程具备以用户为中心的 CIS 特征。用基于 SPDM 模型对实际的申报流程进行电子化，可以得到图 6 所示的流程图(图 6 中的约束信息未标出，且该流程中的约束为柔性约束)。

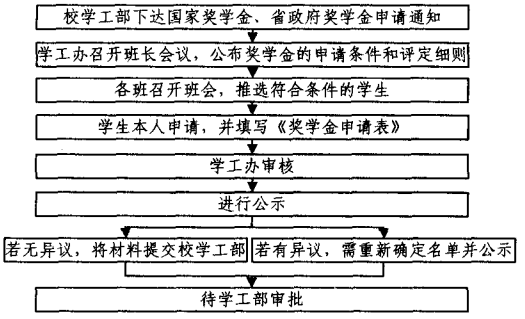

图 5 奖学金申请流程示意图

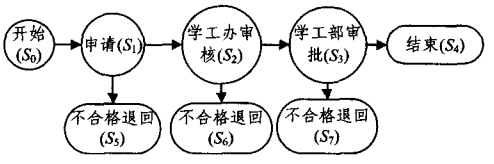

图 6 奖学金申请状态变换图

表单 F(申请表)在流程(P)的推进过程中，随着状态的变化完成审批流程。申请表中数据项较多，有：姓名、性别、出生年月、照片、民族、政治面貌、入学时间、身份证号码、联系电话、大学(学院)、系、班、曾获何种奖励、家庭户口、家庭人口总数、家庭月总收入、人均月收入、收入来源、家庭住址、邮政编码、学习成绩、申请理由、学工办审核意见、学工部审核意见。显然，申请表中有一些数据项不具有约束效力，从而不能作为约束项。设从 S_1 到 S_2 的约束信息为有两个，C_1："家庭户口＝'农业'∧人均月收入＜300∧学习成绩＞＝'中'"，阈值 w_1：0.8；C_2："家庭户口＝'非农'∧人均月收入＜400∧学习成绩＞＝'中'"，阈值 w_2：0.9。其中属性家庭户口的取值范围为{非农，农业}，属性学习成绩的取值范围为{优，良，中，差}，序关系为"优＞良＞中＞差"，且属性学习成绩的取值经由主干课程分数平均值换算而来，不再详细介绍。

按照柔性约束模型，结合系统约束信息，对 C_1，C_2 的约束项中的属性排序及加权，得到结果如下：

1) 人均月收入 0.4；

2) 学习成绩 0.33；

3) 家庭户口 0.27。

系统在运行时，随着表单的进入，将会产生众多流程的例，以表单的编号作为流程和表单的唯一标识，见表1。

表1 奖学金申请表单信息举例

编号	人均月收入	学习成绩	家庭户口
001	200	优	农业
002	350	优	非农
003	380	中	农业
004	400	优	农业
005	300	差	非农
……	……	……	……

系统根据约束信息处理上述表单信息时，分别采取不同的操作，步骤如下。对于表单001，满足约束 C_1，直接提交，进入状态 S_2；表单002，满足约束 C_2，直接提交，进入状态 S_2。表单001和002由于满足现有约束，不经过FCA处理，不再讨论。对于表单003，不满足 C_1 或 C_2，则需要经过FCA处理。分别计算表单003与 C_1 和 C_2 之间的距离，分别用 D_{c1} 和 D_{c2} 表示。

$D_{c1}=1-(0.4*(380-300)/380)=0.92$

$D_{c2}=1-(0.27*1/2)=0.87$

由于 $D_{c1}>D_{c2}$，且 $0.92>0.8$，则可以将表单003提交，状态变为 S_2，待由学工办审批。若 $Reward(003)<0$，即审批未通过，此时需要修改 w_1，$w_1=(1+0.92)/2=0.96$。若 $Reward(003)>0$，即审批通过，此时亦需要修改 w_1，$w_1=0.8-|0.8-0.92|/10=0.79$，其中10为参数 l 的初始值。对于后续的表单处理，可以分两种情况处理。

(1)假设表单003审批通过。系统处理表单004。由于表单004不满足 C_1 和 C_2，则经过FCA对004进行计算。分别计算表单004与 C_1 和 C_2 之间的距离，

$D_{c1}=1-(0.4*(400-300)/400)=0.9$

$D_{c2}=1-(0.27*1/2)=0.87$

由于 $D_{c1}>D_{c2}$，且 $0.9>0.79$，则可以将表单004提交，状态变为 S_2，待由学工办审批。

(2)假设表单003审批未通过。同理，分别计算表单004与 C_1 和 C_2 之间的距离，

$D_{c1}=1-(0.4*(400-300/400)=0.9$

$D_{c2}=1-(0.27*1/2)=0.87$

由于 $D_{c1}>D_{c2}$，但由于 $0.9<0.96$，则此时将表单004退回，进入状态 S_5，不能提交至 S_2(学工办审批)。

对于约束 C_2 的阈值 w_2 的处理方式与上面步骤相同。

剩余表单的处理按照上述方式进行。

从以上过程我们可以看出，在对表单处理的过程中，约束对应的阈值在不停地发生变化(或快或慢)，在一定程度上能够反映用户的意图，也即能够从用户的行为上进行学习，适应环境的变化。

结束语 本文首先对约束的作用及现状进行介绍，说明了引入柔性约束的重要性；然后以UCCIS为研究对象，利用机器学习中的若干技术，在研究对象中引入具有进化功能的柔性约束模型；重点对柔性约束模型中的关键部件——柔性约束适配器FCA的工作原理及流程进行说明，并对FCA产生的日志如何标注为学习样本进行介绍，最后讨论了如何将强化学习引入到FCA以实现约束信息阈值的动态更新。

在CIS中引入柔性约束模型，将会使约束信息变得更为灵活和智能，且能随着环境变化而变化。该模型仅适用于UCCIS，此外，还包括以下几种情况：(1)CIS中的约束关系描述不清晰，约束的属性及取值具有模糊性，因此其是个较为粗略的约束；(2)在CIS的使用过程中约束关系可能发生更改。对于需求信息明确、约束清晰且不接受柔性的CIS系统，无需引入柔性约束模型，否则将会破坏已有约束信息的严肃性和现有系统的稳定性。同理，在其他系统中引入柔性约束模型也需要考虑这些问题，这也是机器学习应用的条件之一。

实验说明本文介绍的模型是有效的，能够增加业务流程的适应性及为用户带来方便，但是仍然存在一些问题需要解决。进一步的研究将在约束信息阈值调整的基础上研究属性的变化情况，即属性的权重能够随着环境的变化而变化，另外，本文算法仅考虑表单状态的一步变化，还需要结合表单的整个流程对不同的约束信息的阈值进行调整。

参考文献

[1] 黎巍，李明树，王青，等. 一种用于软件过程建模的适应性Agent协商[J]. 软件学报，2009，20(3)：557-566

[2] Cugola G, Ghezzi C. Software processes: A retrospective and a path to the future[J]. Software Process—Improvement and Practice, 1998, 4(2): 101-123

[3] Phongpaibul M, Koolmanojwong S, Lam A, et al. Comparative experiences with electronic process guide generator tools[C]// Wang Q, et al., eds. Proc. of the ICSP 2007 Software Process Dynamics and Agility. LNCS 4470, Berlin, Heidelberg: Springer-Verlag, 2007: 61-72

[4] Yilmaz L, Phillips J. Organization-Theoretic perspective for simulation modelling of agile software processes[C]// Wang Q, et al., eds. Proc. of the SPW/ProSim 2006. LNCS 3966, Berlin, Heidelberg: Springer-Verlag, 2006: 234-241

[5] Turetken O, Demirors O. An approach for decentralized process modeling[C]// Wang Q, et al. eds. Proc. of the ICSP 2007 Software Process Dynamics and Agility. LNCS 4470, Berlin, Heidelberg: Springer-Verlag, 2007: 195-207

[6] 陆汝钤，金芝. 从基于知识的软件工程到基于知件的软件工程[J]. 中国科学E辑：信息科学，2008，38(6)：843-863

[7] 李明树，杨秋松，翟健. 软件过程建模方法研究[J]. 软件学报，2009，20(3)：524-545

[8] 吴步丹，金芝，赵彬. 面向服务的建模：一种全过程复用的方法[J]. 计算机学报，2008，31(8)：1293-1308

[9] 姚莉，张维明，王长缨，等. 基于多智能体的复杂信息系统开发方法研究[J]. 管理科学学报，2002，5：44-54

[10] 王琨，袁峰，周利华. 复杂信息系统模型研究[J]. 计算机科学，2006(1)：119-123

[11] 郑云翔. 基于状态的业务流程描述模型及其应用研究[D]. 广州：中山大学，2007

[12] 毛新军，屈婷婷，王戟. 自适应多Agent系统的面向Agent软件开发方法学ODAM[J]. 计算机研究与发展，2008，45(11)：1892-1901

[13] 毛新军，常志明，王戟，等. 面向Agent的软件工程：现状与挑战[J]. 计算机研究与发展. 2006，43(10)：1782-1789

[14] 李晓峰，冯登国，陈朝武，等. 基于属性的访问控制模型[J]. 通信学报，2008，4：90-98

[15] Epshteyn A, Vogel A, DeJong G. Active Reinforcement Learning[C]// Proceedings of the 25th International Conference on Machine Learning. 2008: 296-303

[16] 黄炳强. 强化学习方法及其应用研究[D]. 上海：上海交通大学，2007

[17] 王皓，高阳，陈兴国. 强化学习中的迁移：方法和进展[J]. 电子学报，2008，12：39-43